



Brain Signal Acquisition

Course S2022

Faculty: Dr Annushree Bablani

Background

- ❖ The brain communicates using spikes-- produced when the neuron receives enough input current from other neurons via synaptic connections.
- ❖ Recording brain activity are based on detecting changes in electrical potentials in neurons
 - ❖ invasive techniques based on implanting electrodes
- ❖ Or on detecting changes in large populations of neurons
 - ❖ noninvasive techniques such as electroencephalography or EEG

Recording signals from brain

Invasive Approaches:

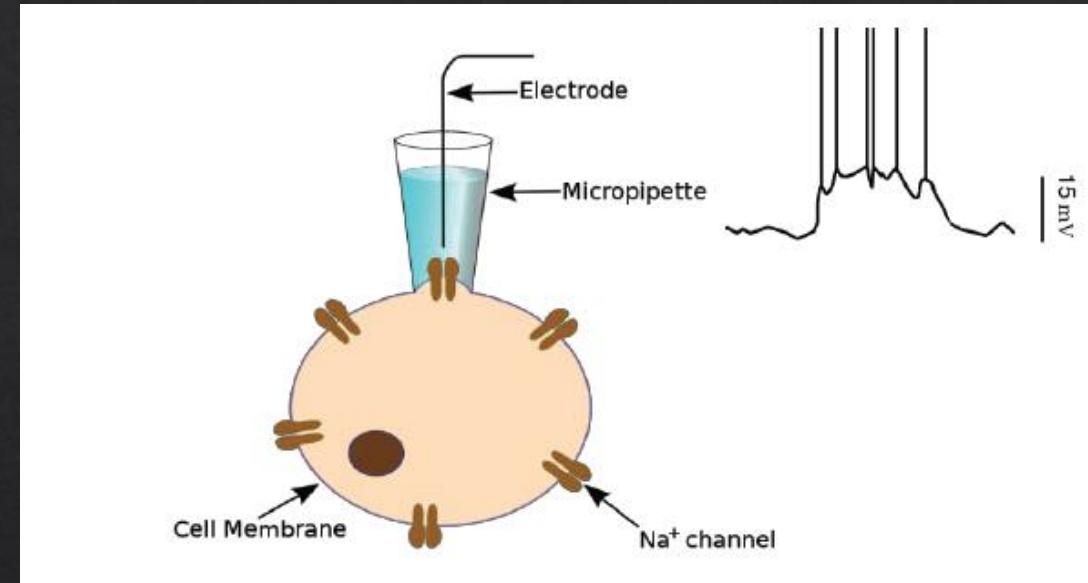
- ❖ Techniques that allow recording from individual neurons in the brain are typically invasive.
- ❖ They involve some form of surgery,
 - ❖ A part of the skull is removed, an electrode or implant placed in the brain, and the removed part of the skull then replaced.
- ❖ A major advantage of invasive recordings is that they allow recording of action potentials at the millisecond timescale.

Invasive Approaches

- ❖ Microelectrodes:
- ❖ A *microelectrode* is simply a very **fine wire** or other **electrical conductor** used to make contact with brain tissue.
- ❖ A typical electrode is made of **tungsten or platinum- iridium alloy** and is insulated except at the tip, which measures around $1\mu\text{m}$ in diameter (A neuron's cell body diameter is in the range of tens of μm).

Invasive Approaches

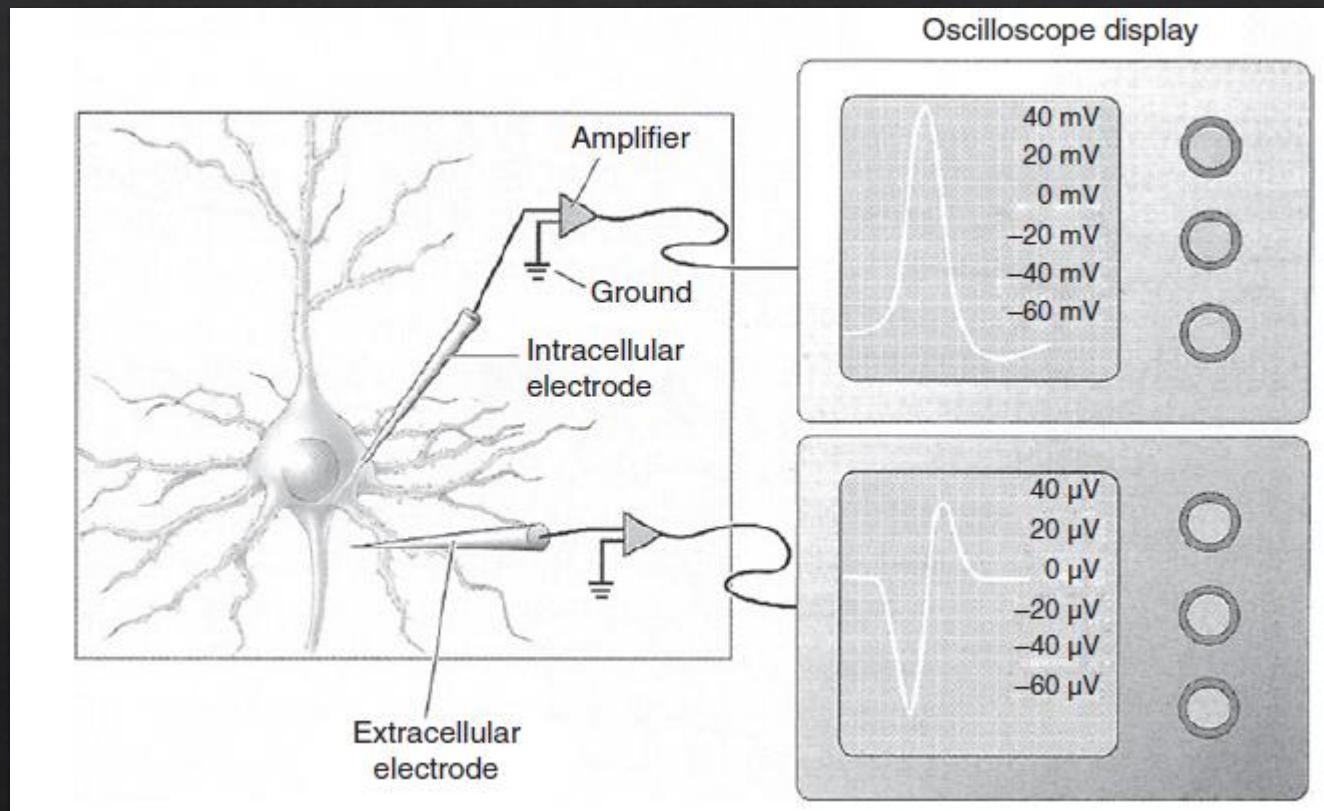
- ❖ Intracellular Recording:
- ❖ Measures the voltage or current **across the membrane of the neuron.**
- ❖ The most common technique, known as *patch clamp recording*.
- ❖ Very Delicate → Intracellular recordings are typically performed only on **slices of brain tissue**



Invasive Approaches

- ❖ Extracellular Recording:
- ❖ Recording of a **single neuron** (or single“unit”): a tungsten or platinum-iridium microelectrode with a tip size of less than 10 microns is inserted into the target brain area.
- ❖ The magnitude of the recorded signal is usually less than a millivolt and thus requires the use of amplifiers to detect the signal.
- ❖ The signal from the amplifier is fed to a computer, which performs additional processing such as filtering noise and isolating the spikes (action potentials).

Invasive Approaches



(from Bear et al., 2007).

Invasive Approaches

- ❖ When the neuron produces a spike, **positive ions flow away** from the extracellular electrode into the neuron, causing the **initial negative deflection** in the display. This is **followed by a positive deflection** as the action potential decreases and **positive charges flow out** of the neuron toward the extracellular electrode.

Invasive Approaches

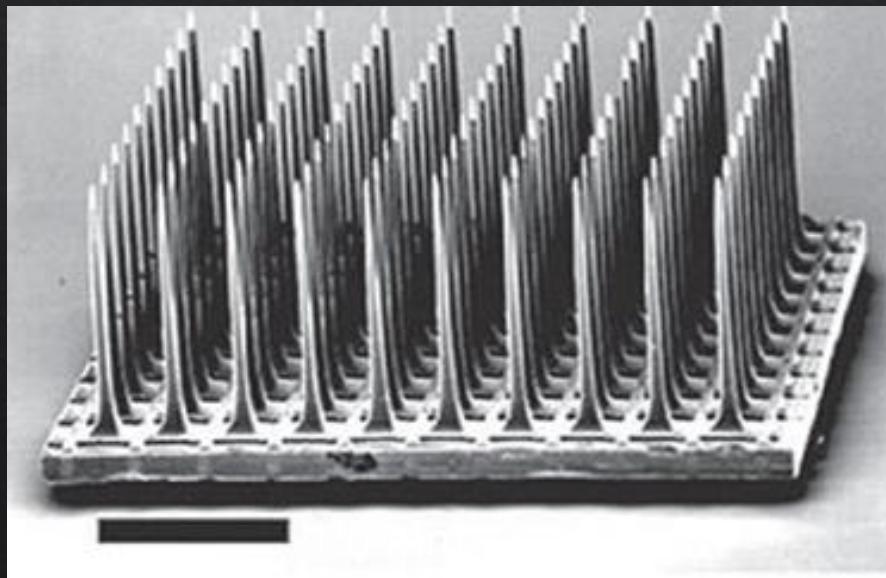
Tetrodes and Multi-Unit Recording:

- ❖ To record from **multiple neurons simultaneously** by using more than one electrode.
- ❖ **Four wires** are tightly wound together in a bundle.

Invasive Approaches

Multielectrode Arrays:

- ❖ To record from larger numbers of neurons, microelectrodes can be arranged in a **grid-like structure** to form a **multielectrode array** of $m \times n$ electrodes.



(adapted from Hochberg et al., 2006).

Invasive Approaches

- ❖ The most common types of implantable arrays are microwire, silicon-based, and flexible microelectrode arrays
- ❖ Increased **spatial resolution**
- ❖ The ability to record simultaneously from several dozens of neurons
- ❖ Opens the door to extracting complex types of information such as position or velocity signals that could be useful for controlling prosthetic devices.

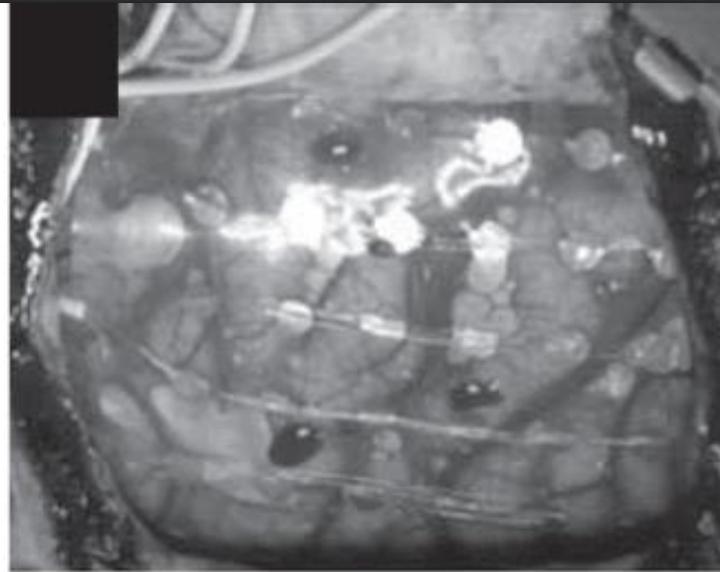
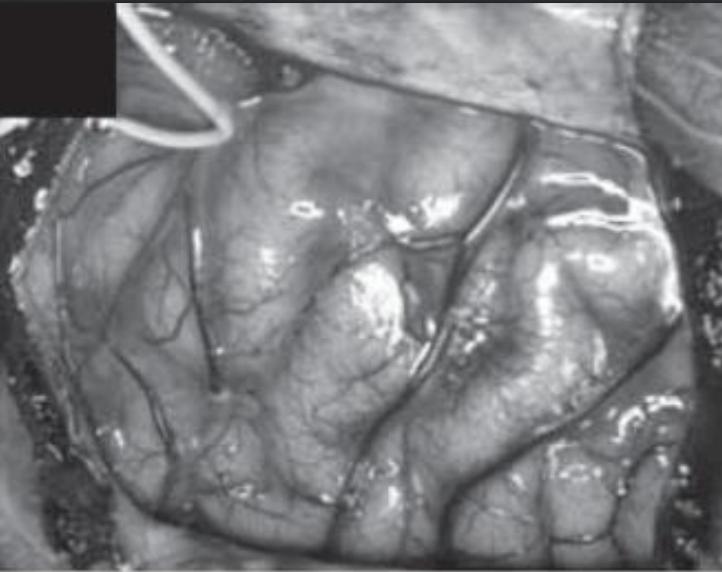
Partially Invasive Approach

Electrocorticography (ECoG):

- ❖ *Electrocorticography (ECoG)* is a technique for recording brain signals that involves **placing electrodes on the surface of the brain**.
- ❖ The procedure requires making a surgical incision into the skull to implant the electrodes on the brain surface

Partially Invasive Approach

- ❖ ECoG electrodes can record the electrical fluctuations caused by the **coherent activity of large populations of neurons** (several tens of thousands).
- ❖ **Safer** than arrays implanted inside the brain.
- ❖ ECoG electrodes may also be **less likely to wear out** compared to brain penetrating electrodes
- ❖ ECoG offers greater **spatial resolution**



(from (Miller et al., 2007)).

Partially Invasive Approach

MicroECoG:

- ❖ One disadvantage of ECoG, is the relatively large size of ECoG electrodes
- ❖ These microelectrodes are only a fraction of a millimeter in diameter and spaced only 2–3 mm apart in a grid
- ❖ Allows detection of neural activity at a much finer resolution than traditional ECoG.
- ❖ Decoding fine movements, such as the movements of individual fingers, or even speech, without actually penetrating the brain.

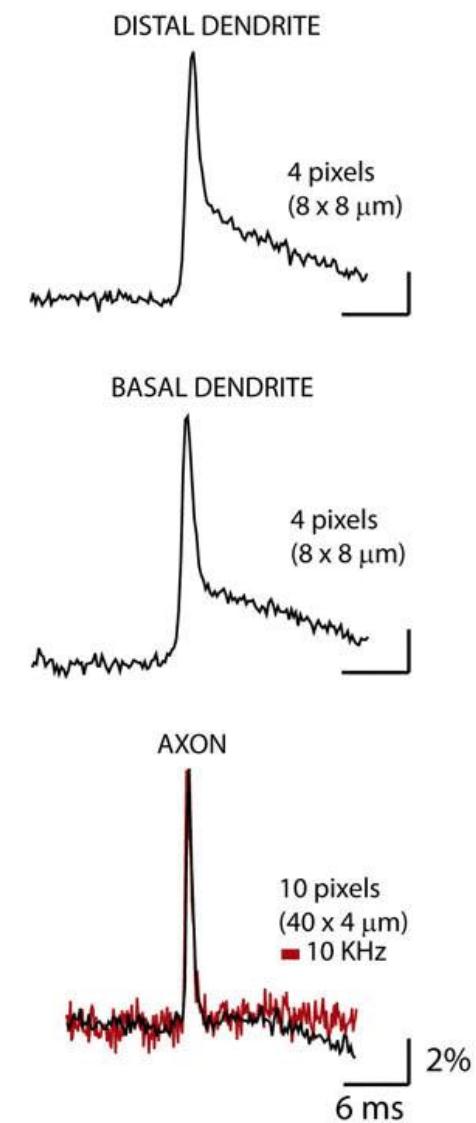
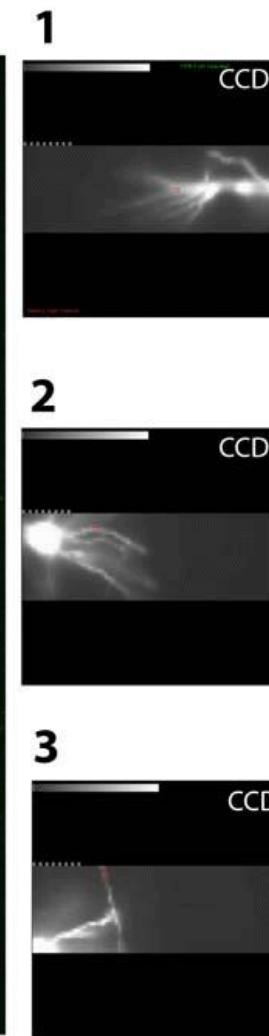
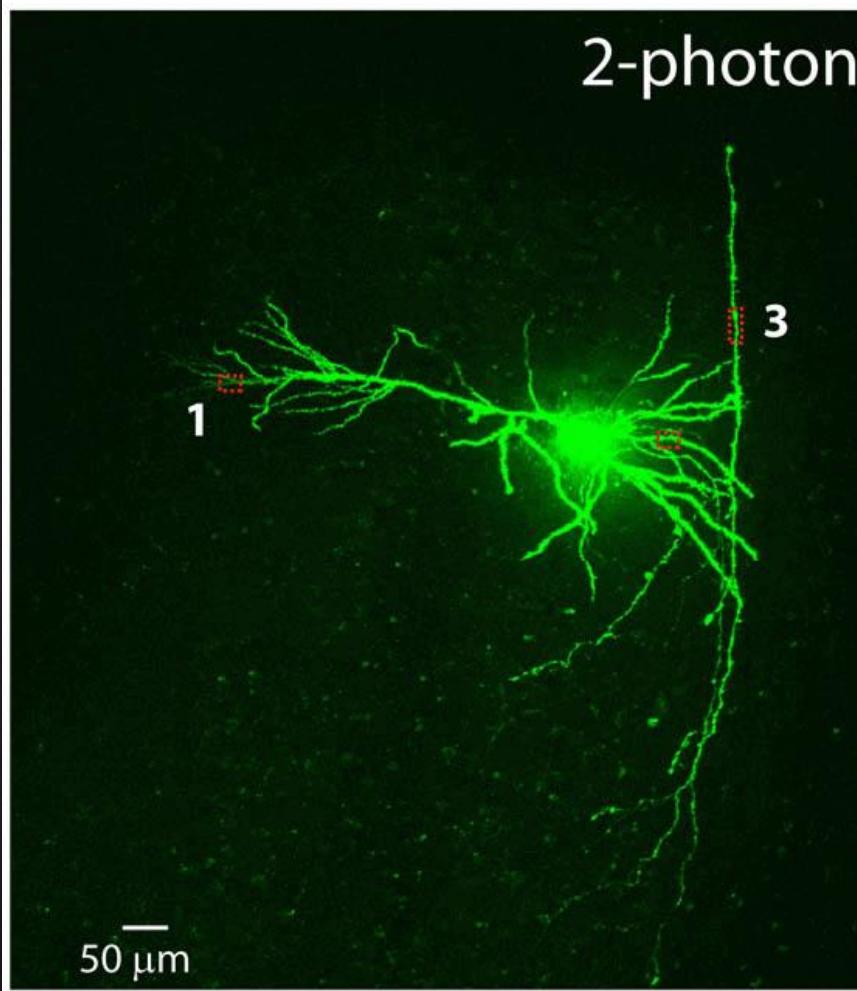
Partially Invasive Approach

Optical Recording: Voltage-Sensitive Dyes and Two-Photon Calcium Imaging:

- ❖ **Voltage-sensitive dyes**
- ❖ Neurons are stained with a voltage-sensitive dye
- ❖ Dye responds to changes in membrane potential by changing its absorption and/or fluorescence
- ❖ Recorded optical signals correspond to **summed responses** from several **simultaneously active neurons**.
- ❖ Useful for imaging macroscopic features of the brain such as feature maps in the cortex

VOLTAGE-SENSITIVE DYE IMAGING

SINGLE TRIAL RECORDINGS AT 5 KHz



(image: Scholarpedia http://www.scholarpedia.org/article/Voltage-sensitive_dye).

Partially Invasive Approach

- ❖ Two-photon calcium imaging
- ❖ Based on the fact that electrical activity in neurons is typically associated with changes in calcium concentration.
- ❖ Photon calcium imaging involves:
 - (1) using pressure ejection to load neurons with fluorescent calcium-indicator dyes
 - (2) monitoring changes in calcium fluorescence during neural activity using two-photon microscopy.

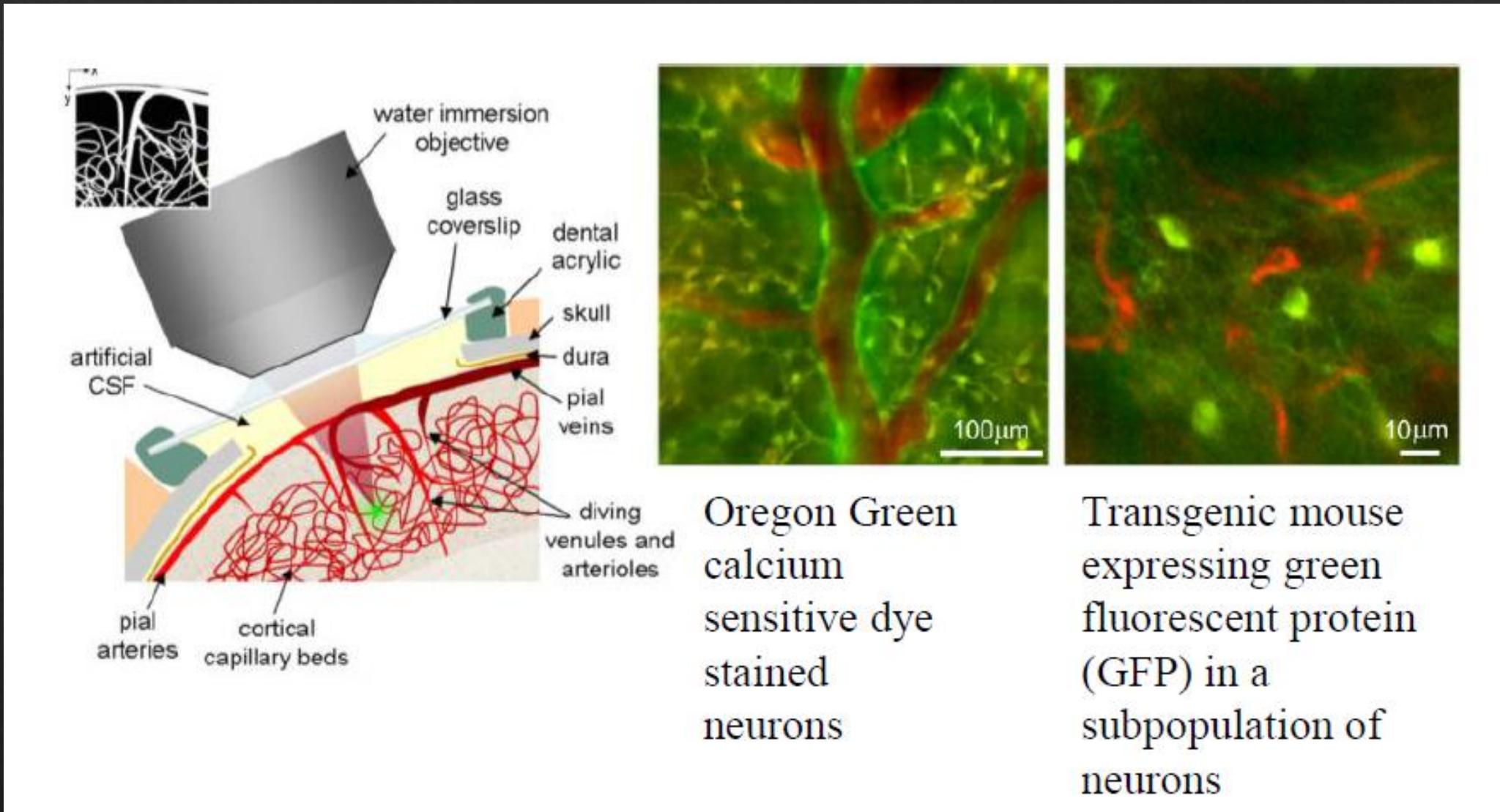


Image from Kherlopian et al., 2008).

Non Invasive Approaches

Electroencephalography (EEG)

- ❖ EEG signals reflect the summation of postsynaptic potentials from many thousands of neurons that are oriented radially to the scalp.
- ❖ EEG predominantly captures electrical activity in the cerebral cortex, whose columnar arrangement of neurons and proximity to the skull favor recording by EEG.

Non Invasive Approaches

- ❖ Electroencephalography (EEG)
- ❖ The spatial resolution is typically poor (in the square centimeter range)
 - ❖ Due to lots of muscles between the source of signal and the electrodes placed on the scalp.
- ❖ The temporal resolution is good (in the milliseconds range)

Non Invasive Approaches

- ❖ The measured signals are in the range of a few tens of microvolts, necessitating the use of powerful amplifiers and signal processing to amplify the signal and filter out noise.
- ❖ Artifacts in the EEG signal
 - ❖ eye movements, eye blinks, eyebrow movements, talking, chewing, and head movements

Non Invasive Approaches

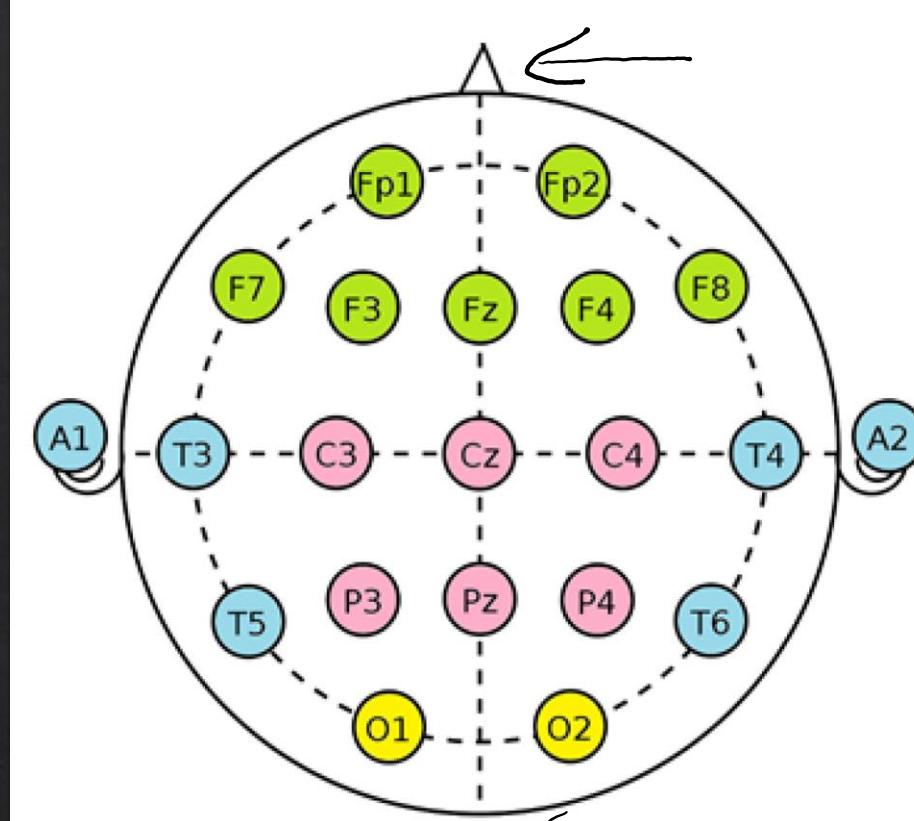
- ❖ EEG recording involves the subject wearing a cap or a net into which the recording electrodes are placed
- ❖ A conductive gel or paste is injected into the holes of the cap before placing the electrodes.



courtesy K. Miller

Non Invasive Approaches

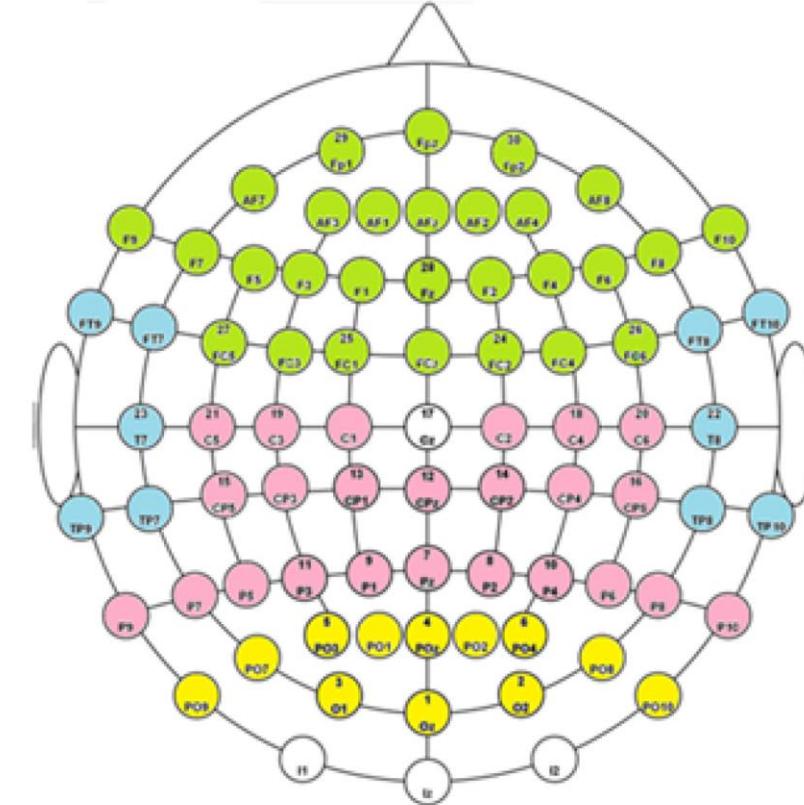
- ❖ The international 10–20 system is a convention used to specify standardized electrode locations on the scalp.
- ❖ C = central, P = parietal, T = temporal, F = frontal, Fp = frontal polar, O = occipital, A = mastoids



10-20 Electrode System



Temporal Lobe



10-10 Electrode System

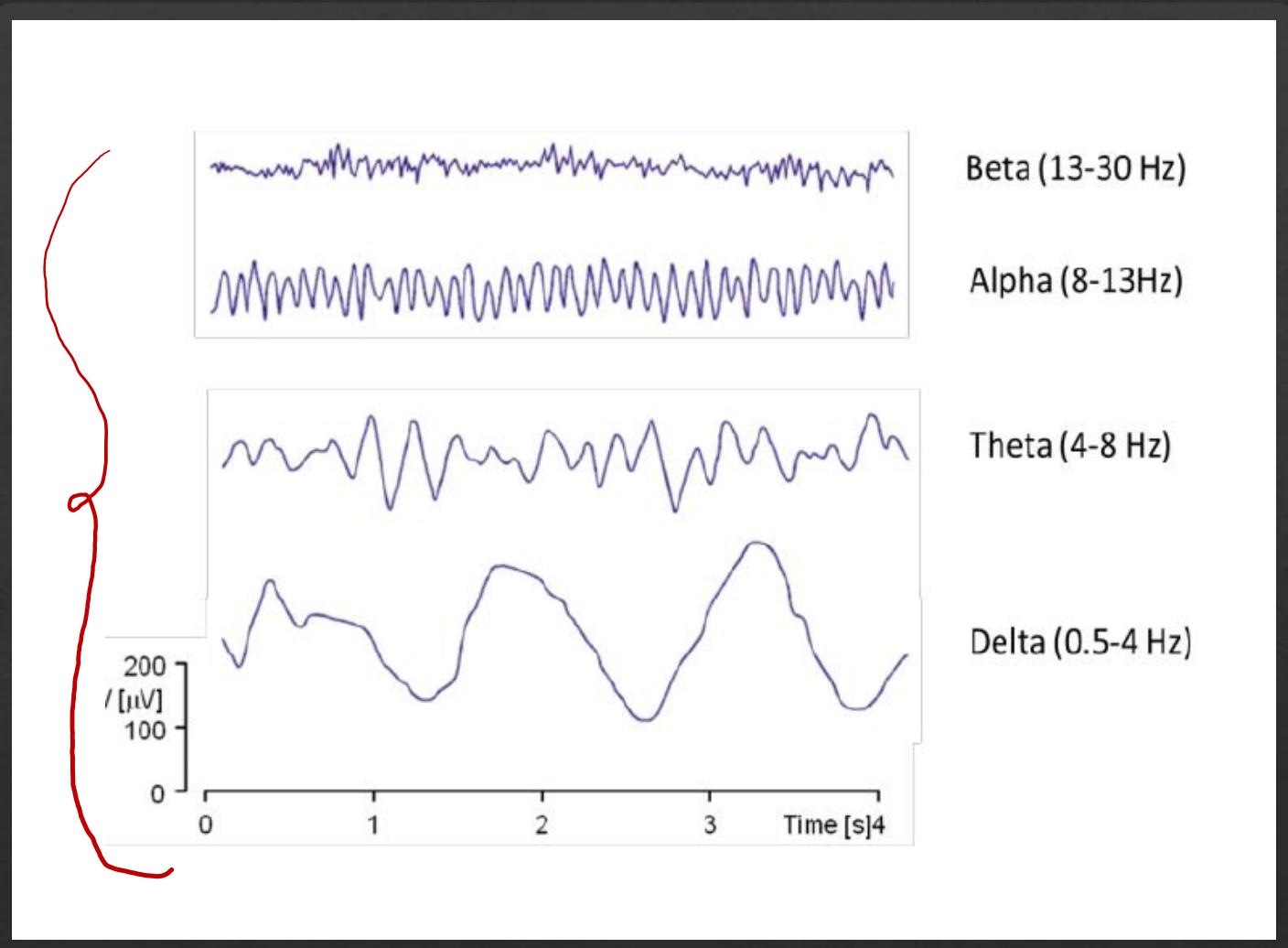
Parietal Lobe

Occipital Lobe

- ❖ The mastoids reference electrode locations behind each ear (A1 and A2).
- ❖ Other reference electrode locations are **nasion**, at the top of the nose, level with the eyes; and **inion**, at the base of the skull on the midline at the back of the head.
- ❖ In a typical setup, each EEG electrode is connected to one input of a differential amplifier, and the other input is connected to a reference electrode

- ❖ The amplification of voltage between the active electrode and the reference is typically 1,000–100,000 times.
- ❖ The amplified signal is passed through a filter and then digitized via an A/D (analog to digital) converter.
- ❖ After digitization, the EEG signal may be additionally filtered by a 1–50 Hz bandpass filter.
 - ❖ Excludes noise and movement artifacts in the very low and very high frequency ranges.

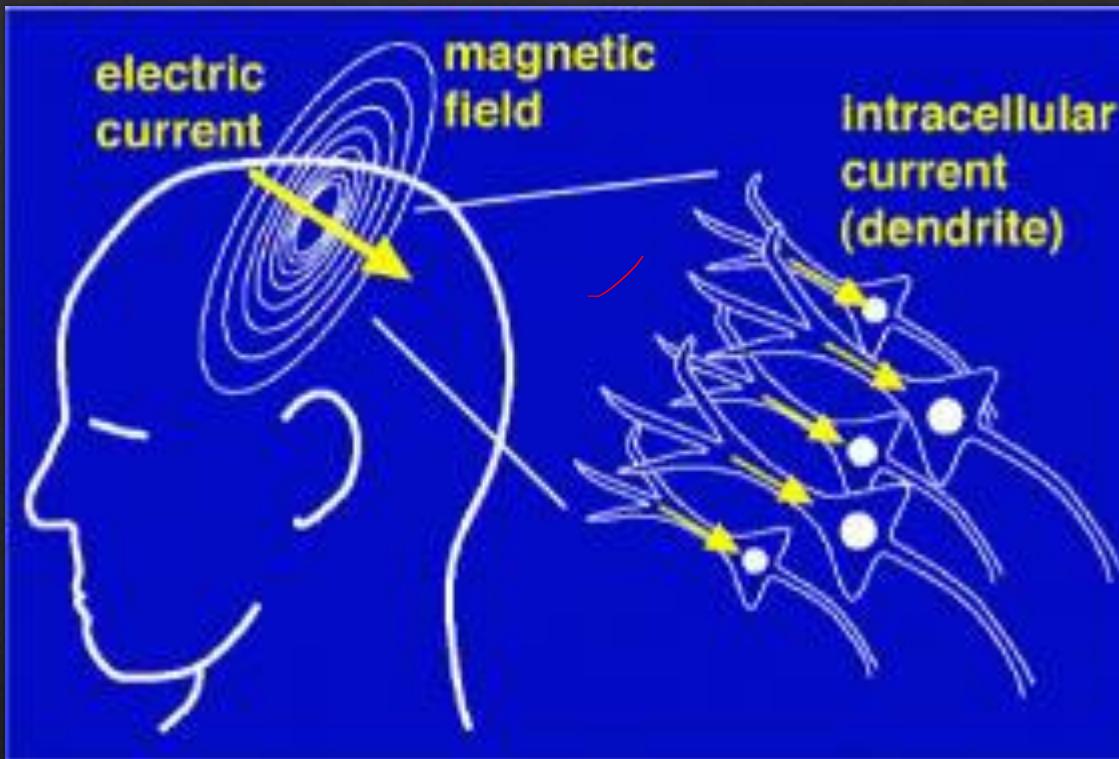
- ❖ EEG recordings are well-suited to capturing oscillatory brain activity or “brain waves” at a variety of frequencies
 - ❖ Alpha waves (8 to 13 Hz)
 - ❖ Beta waves (13 to 30 Hz)
 - ❖ Delta waves (0.5-4 Hz)
 - ❖ Theta waves (4-8 Hz)
 - ❖ Gamma waves (30-100 Hz or more)



Non Invasive Approaches

Magnetoencephalography (MEG):

- ❖ Measures **magnetic fields** produced by activity of thousands of cortical neurons oriented perpendicular to the cortical surface
- ❖ Magnetic fields not distorted by skull and scalp
- ❖ Better **spatial resolution** than EEG
- ❖ Expensive and bulky
- ❖ Magnetically shielded rooms



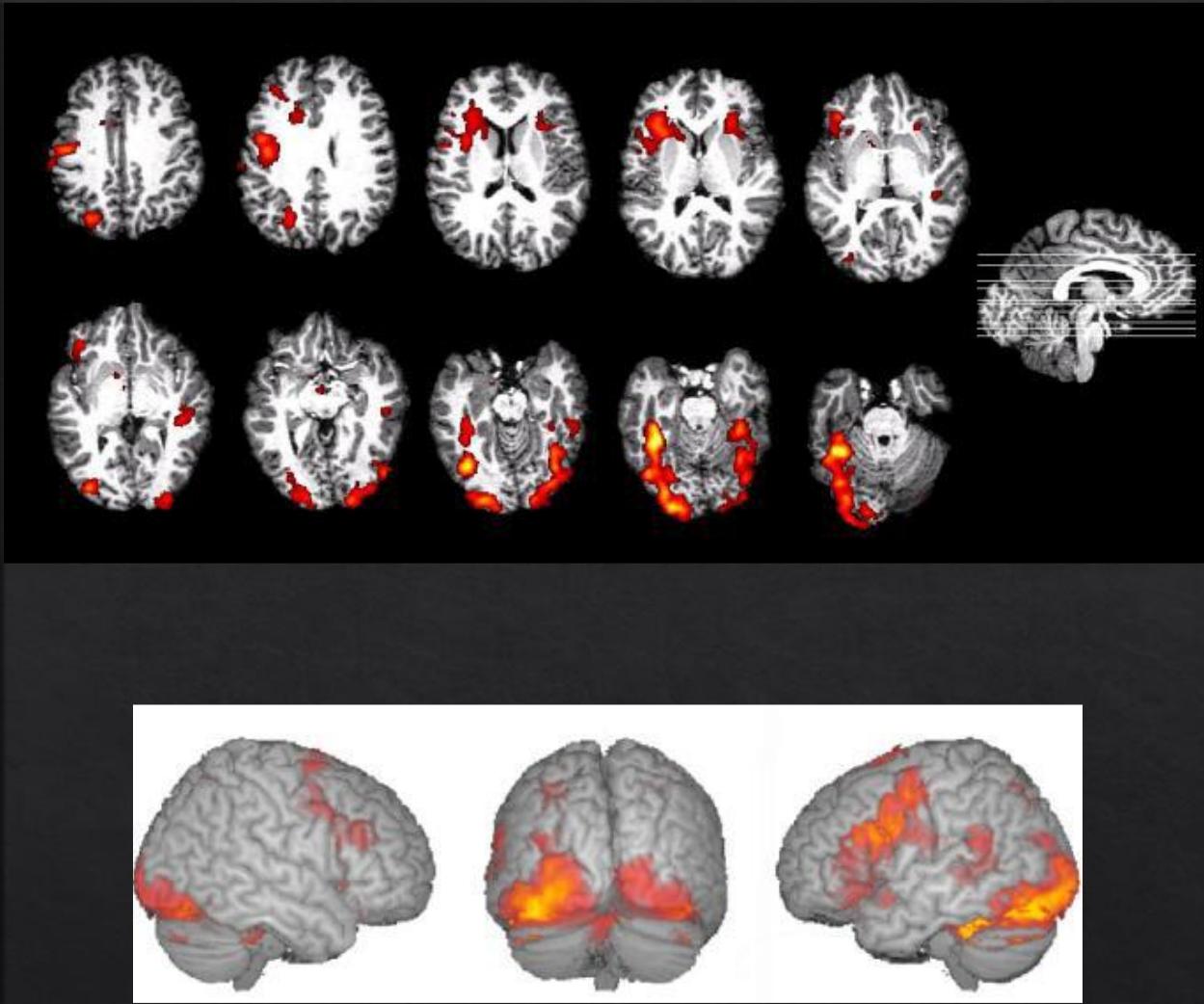
(image A: Wikimedia Commons;

image B: http://dateline.ucdavis.edu/photos_images/dateline_images/040309/DondersMEGOle_W2.jpg).

Non Invasive Approaches

Functional Magnetic Resonance Imaging (fMRI) :

- ❖ Measures **changes in blood flow** due to increased activation of neurons in an area
- ❖ Relies on paramagnetic properties of oxygenated and deoxygenated hemoglobin in the blood
- ❖ Produces images showing **blood-oxygenation-level-dependent** signal changes (BOLD)

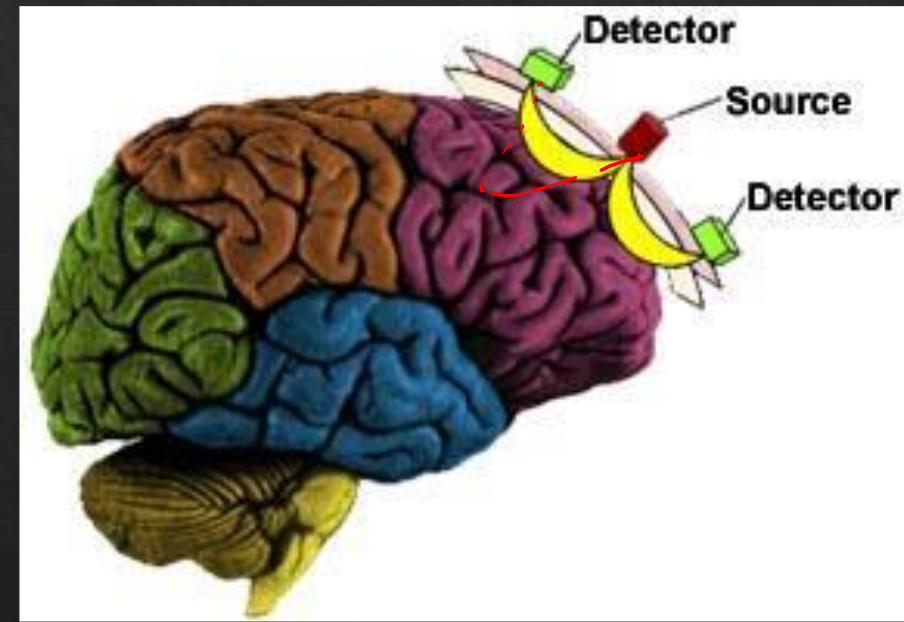
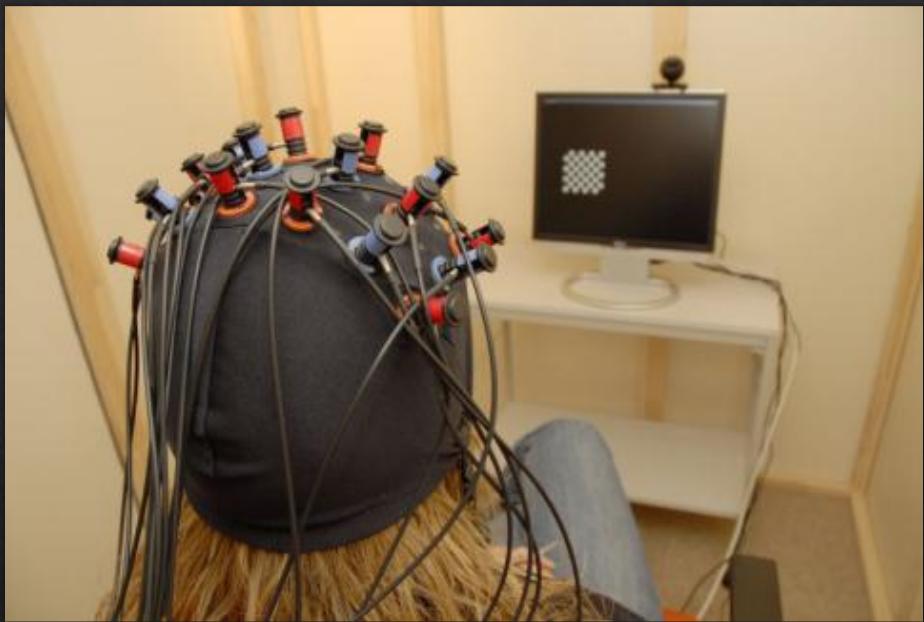


Example fMRI Images (word reading task)

Non Invasive Approaches

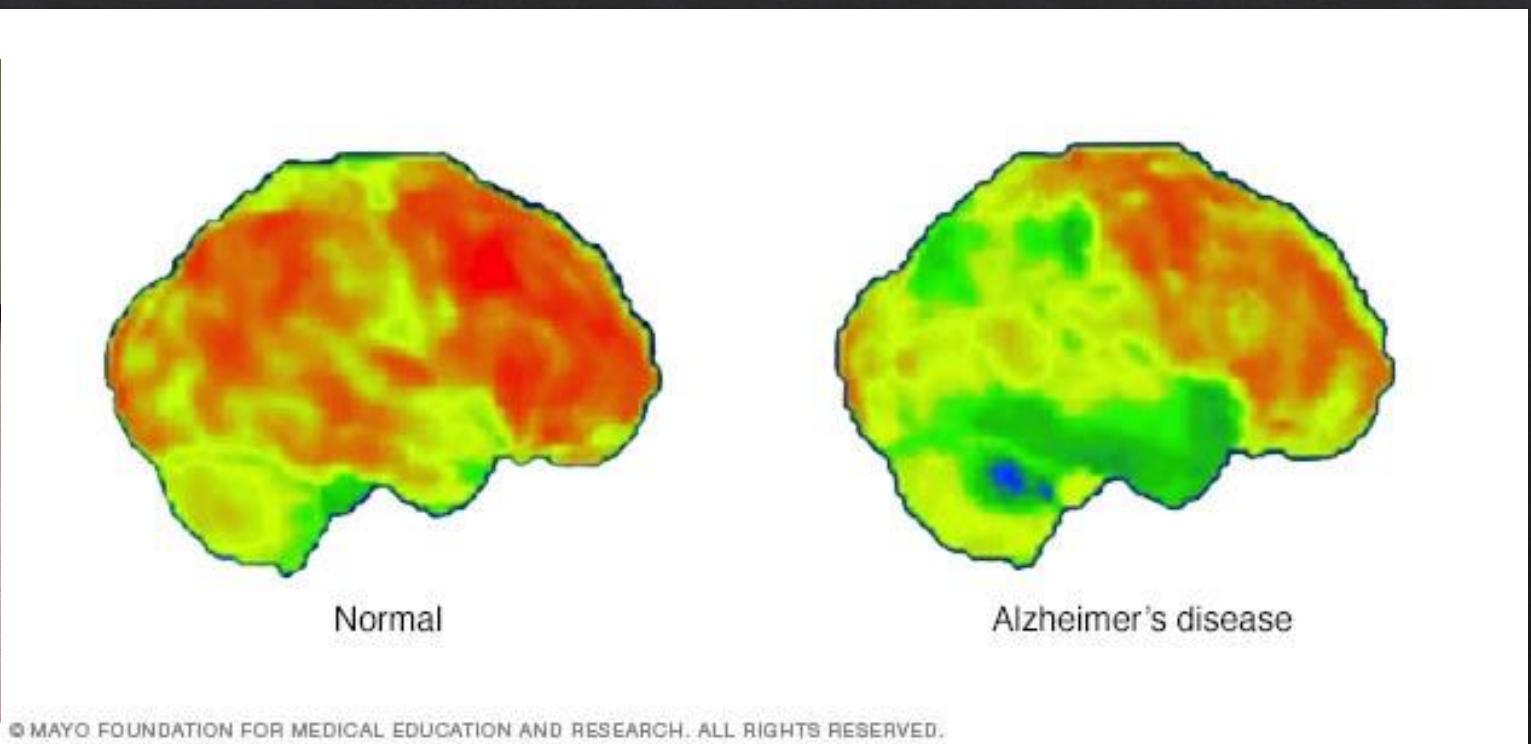
Functional Near-Infrared Spectroscopy (fNIR)

- ❖ Measures change in blood oxygenation level caused by increased neural activity in the brain.
- ❖ Based on detecting **near-infrared light absorbance** of hemoglobin in the blood with and without oxygen.
- ❖ Maps neural activity using “*optodes*”(emitters and detectors)



Non-Invasive Approaches

- ❖ **Positron Emission Tomography (PET):**
- ❖ Measures emissions **from radioactively labeled, metabolically active chemicals** that have been injected into the bloodstream for transportation to the brain.
 - ❖ The labeled compound is called a *radiotracer*.
- ❖ Sensors in the PET scanner detect the radioactive compound
 - ❖ As a result of metabolic activity caused by brain activity.
- ❖ Generate two-or three-dimensional images indicating the amount of brain activity.

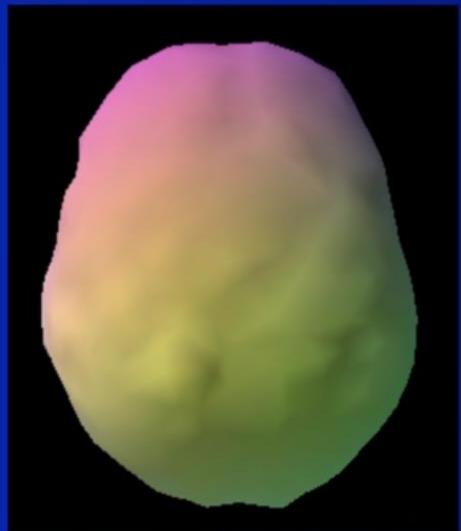


A PET scan can compare a normal brain (left) with one affected by Alzheimer's disease (right). An increase in blue and green colors shows decreased brain metabolic activity due to Alzheimer's disease.

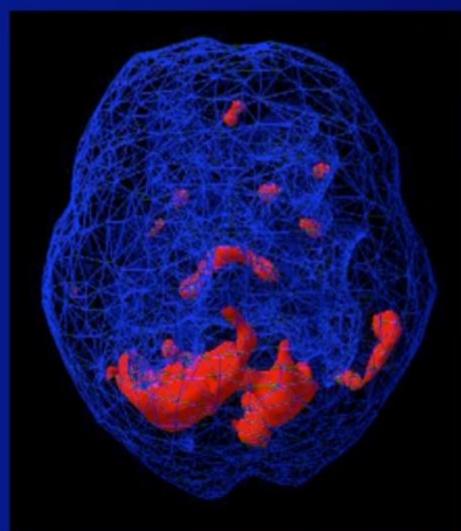
Non Invasive Approaches

- ❖ Single-photon emission-computed tomography (**SPECT**):
- ❖ SPECT is a nuclear medicine technique that uses **gamma rays** to study the brain.
- ❖ A **radioactive substance** is injected into the patient's body and is scanned using a SPECT machine.
- ❖ Allows doctors to see **how blood flows** into tissues and organs.
 - ❖ Active, inactive, or overactive.
- ❖ Averages the brain activity over a few minutes and generates an image.

Healthy Brain SPECT Scans

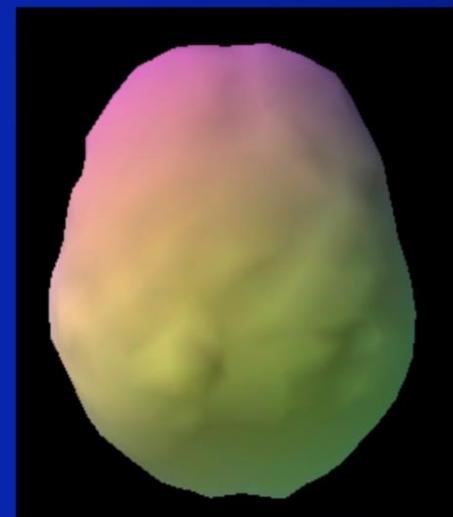


Surface View

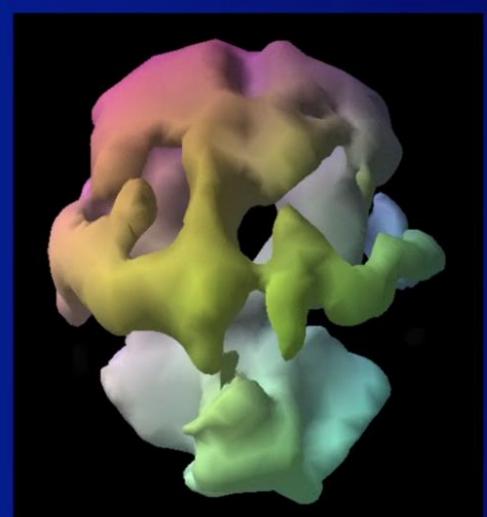


Active View

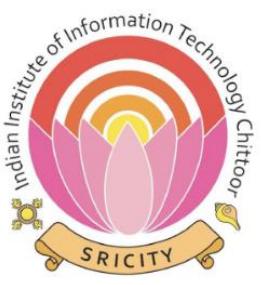
Healthy vs Alzheimer's Disease



Healthy



Alzheimer's



EEG Measure

Course Instructor

Dr. Annushree Bablani

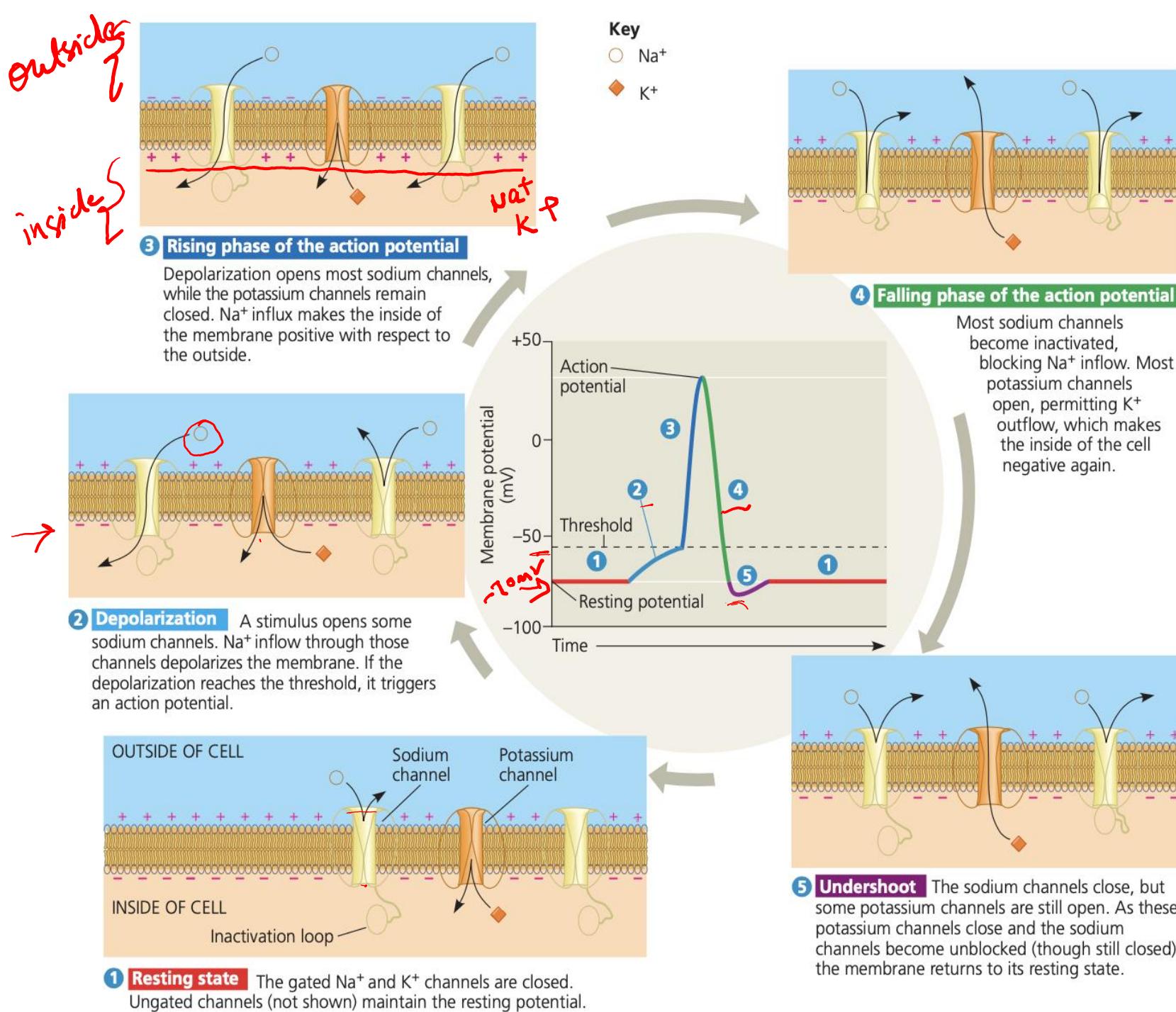
Acknowledgments: Dr Sreeja SR

REVISIT

- **Neurons** are nerve cells that transfer information within the body
- Neurons use two types of signals to communicate: electrical signals (long-distance) and chemical signals (short-distance)
- Nervous systems process information in three stages: sensory input, integration, and motor output

Hyperpolarization and Depolarization

- Changes in membrane potential occur because neurons contain **gated ion channels** that open or close in response to stimuli.
- When gated K⁺ channels open, K⁺ diffuses out, making the inside of the cell more negative. This is **hyperpolarization**, a reduction in magnitude of the membrane potential.
- Opening other types of ion channels triggers a depolarization, an increase in the magnitude of the membrane potential. For example, depolarization occurs if gated Na⁺ channels open and Na⁺ diffuses into the cell.
- **Graded potentials** are changes in polarization where the magnitude of the change varies with the strength of the stimulus.
- If a depolarization shifts the membrane potential sufficiently, it results in a massive change in membrane voltage called an **action potential**.



Generation of Postsynaptic Potentials

- Direct synaptic transmission involves binding of neurotransmitters to **ligand-gated ion channels** in the postsynaptic cell.
- Neurotransmitter binding causes ion channels to open, generating a postsynaptic potential.
- Postsynaptic potentials fall into two categories
 - **Excitatory postsynaptic potentials (EPSPs)** are depolarizations that bring the membrane potential toward threshold.
 - **Inhibitory postsynaptic potentials (IPSPs)** are hyperpolarizations that move the membrane potential farther from threshold.

What are we measuring with EEG?

What does the EEG record?

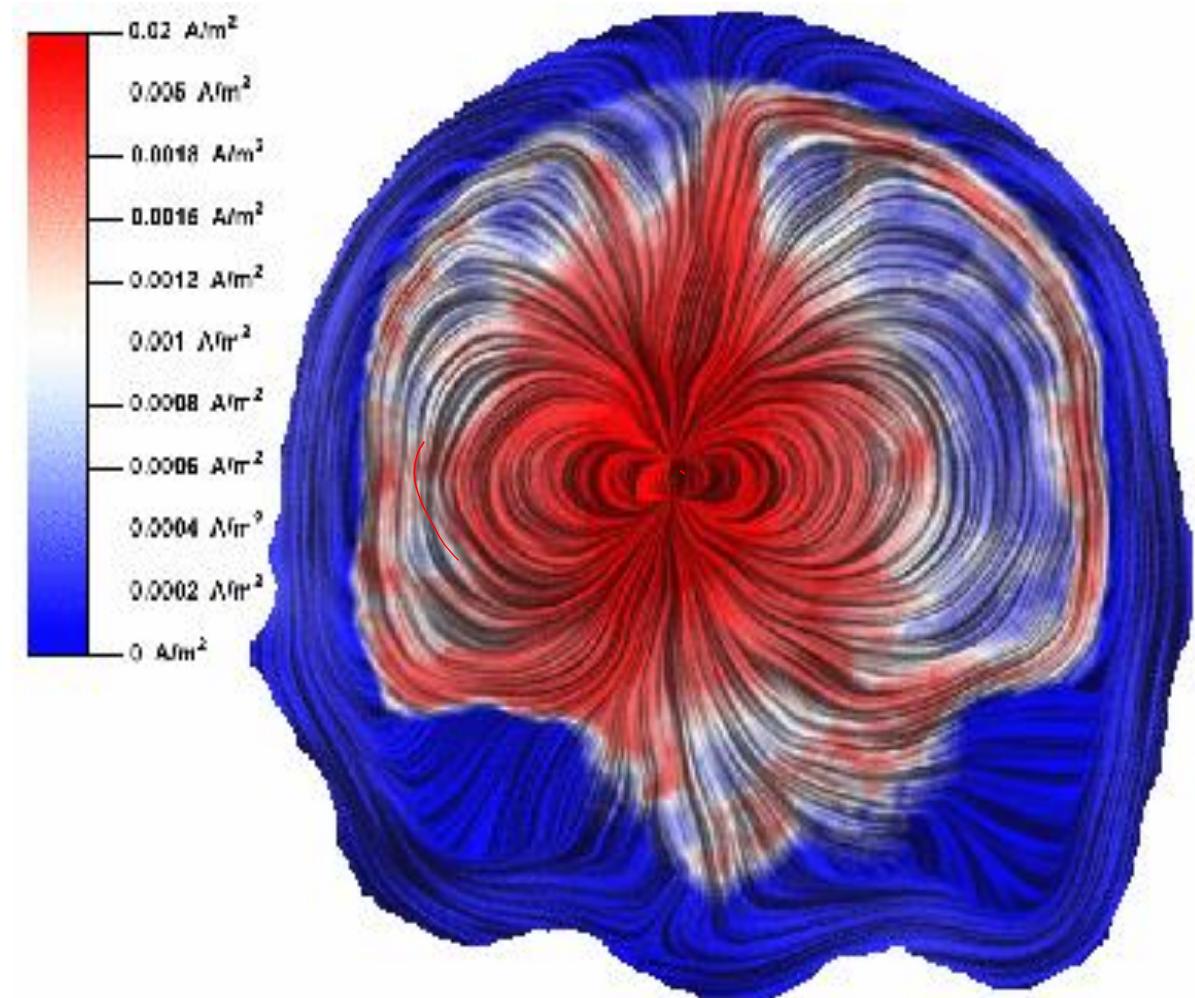
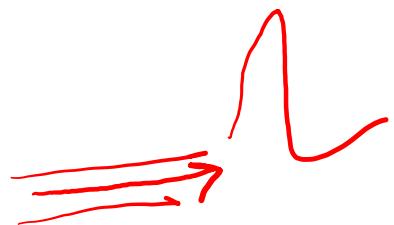
Mainly NOISE!!

Volume Conduction

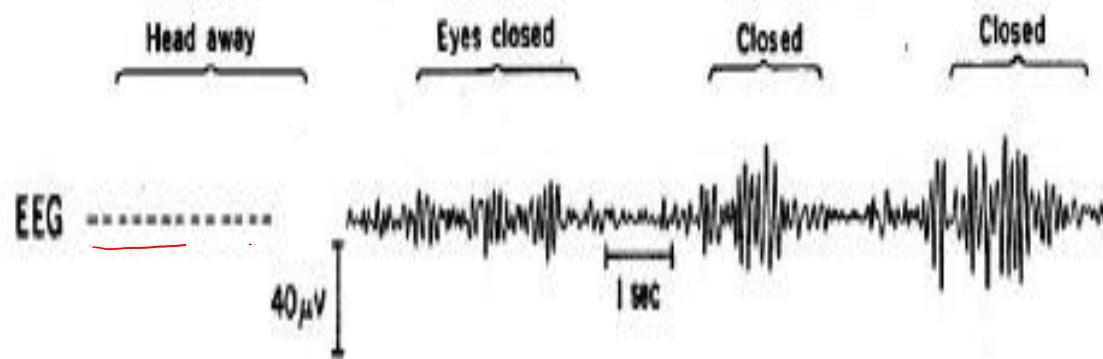
EEG directly measures the neural activity.

What kind of neuronal activity?

- Action potentials
- Postsynaptic potentials

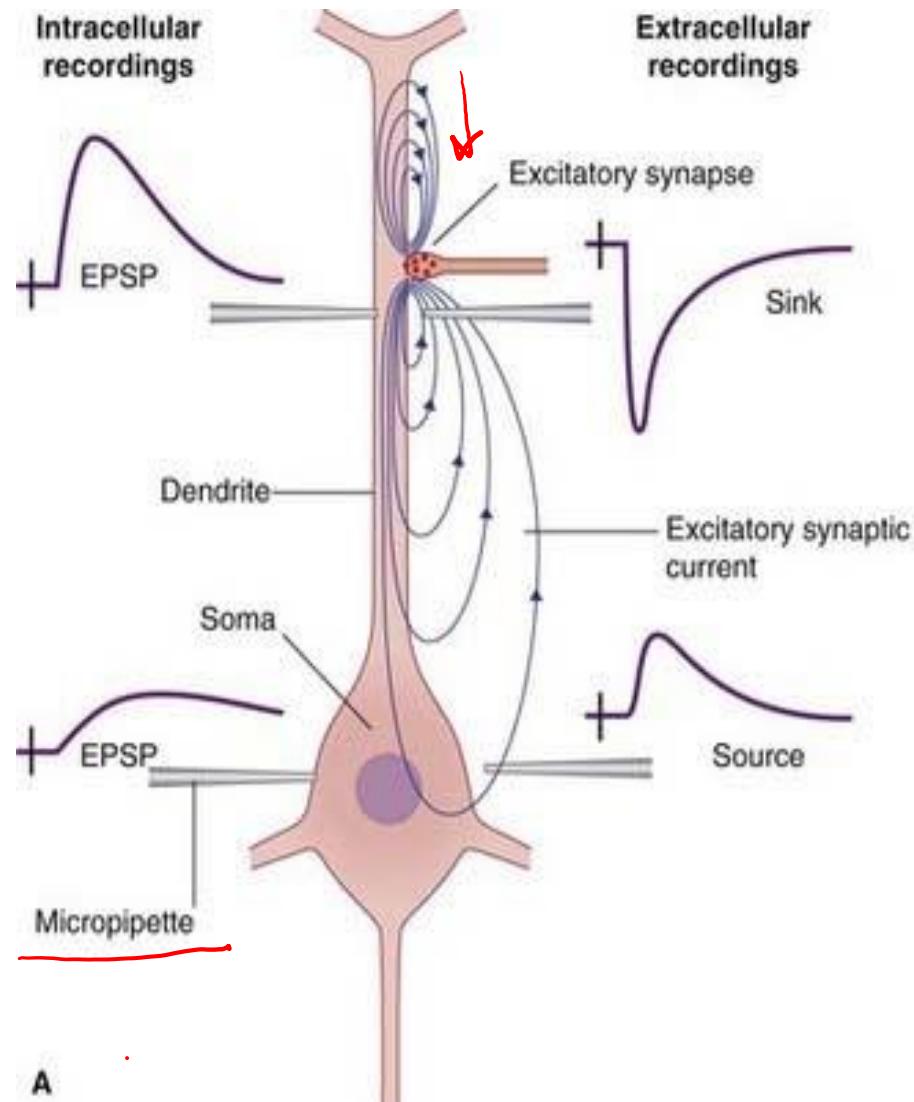


EEG measures the underlying neural currents



- **EEG:** Measures differences in electric potential at the scalp.
- Non-invasive, cover the whole head, and have very high temporal resolution.

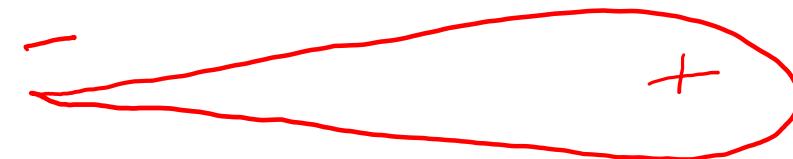
Neural basis of the EEG



When an EPSP is generated in the dendrites of a neuron an extracellular electrode detects a negative voltage difference, resulting from Na⁺ currents flowing inside the neuron's cytoplasm.

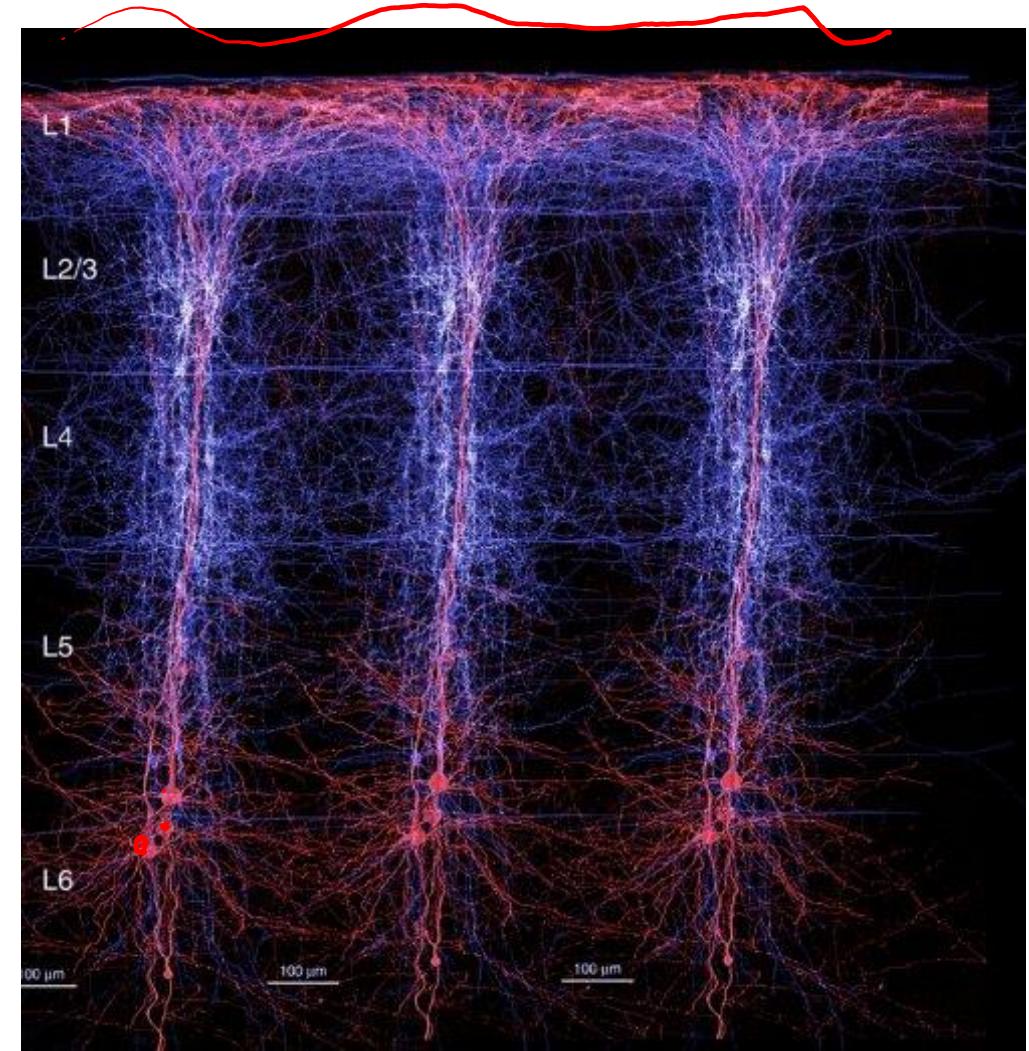
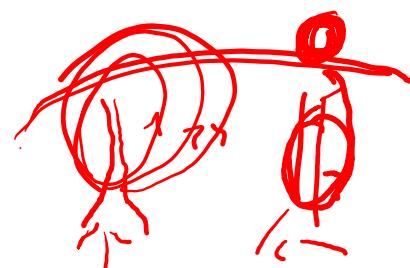
The current completes a loop further away the excitatory input (Na⁺ flows outside the cell), being recorded as a positive voltage difference by an extracellular electrode.

This process can last hundreds of milliseconds.

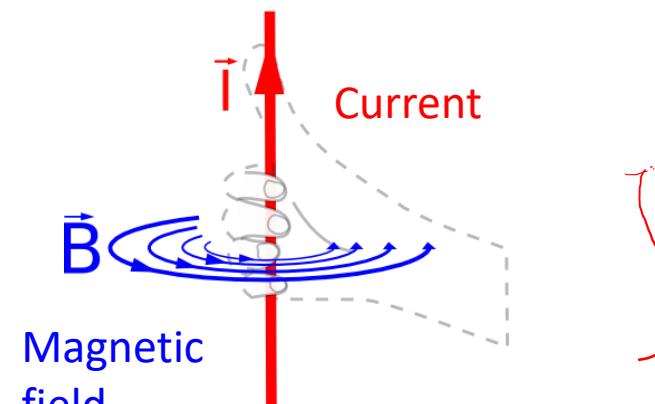
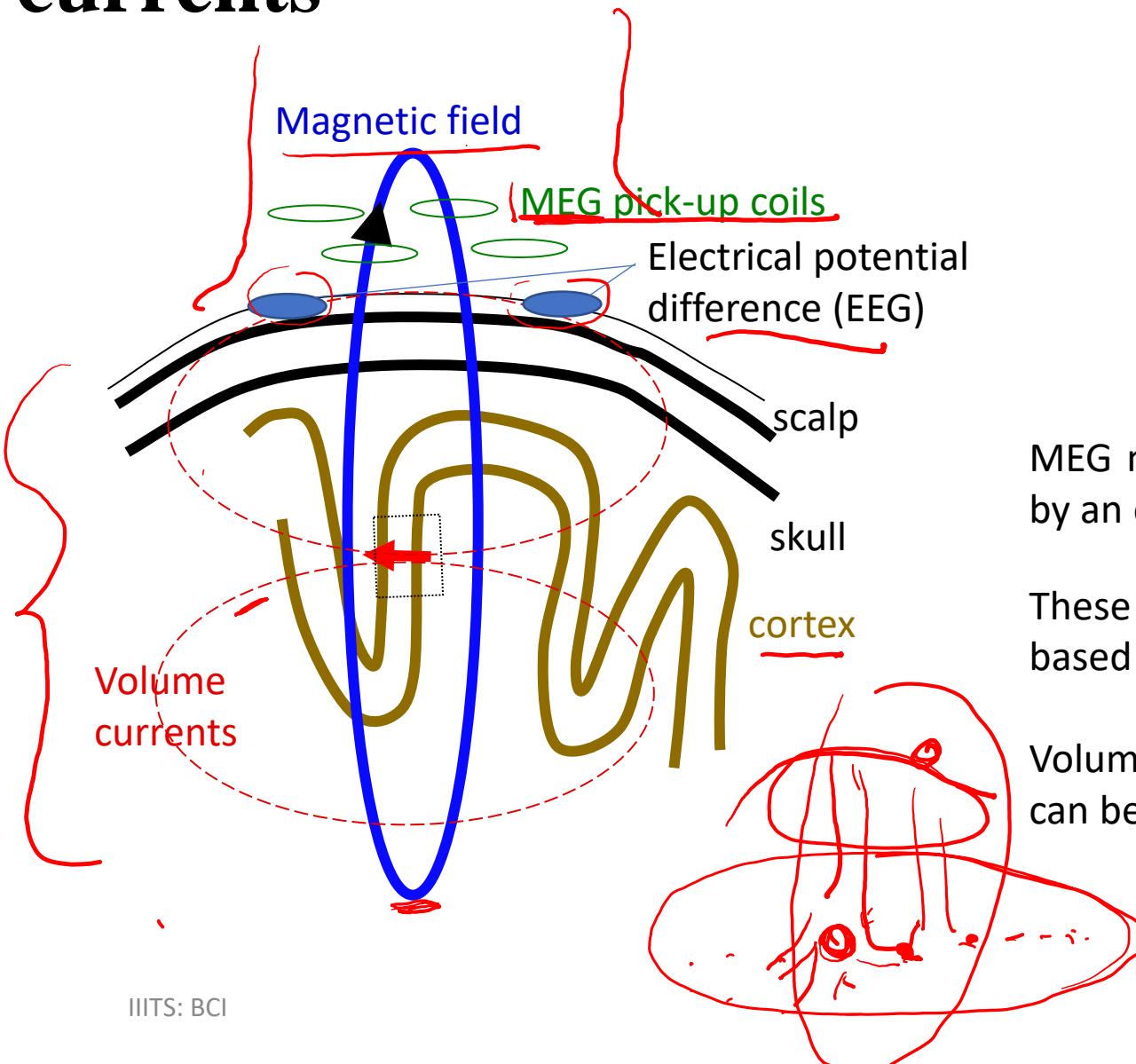


From single neuron to neural population

- The geometry of the neuron must give rise to a net dipole current to contribute to the EEG signal.
The geometry of the neuron must give rise to a net dipole current to contribute to the EEG signal.
- A large number of neurons have to be active simultaneously to generate a measurable EEG signal.
- Pyramidal neurons are spatially aligned and perpendicular to the cortical surface.
Pyramidal neurons are spatially aligned and perpendicular to the cortical surface.
- Thus, EEG represents mainly the postsynaptic potentials of pyramidal neurons close to the recording electrode.
- The electrical activity from deeper generators gets dispersed and attenuated by volume conduction effects.



Primary intracellular currents give rise to volume currents



MEG measures the changes in the magnetic field generated by an electric current (Sarvas 1987, Hämäläinen 1993)

These magnetic fields are mainly induced by primary currents based on excitatory activity (Okada et al. 1997)

Volume currents yield potential differences on the scalp that can be measured by EEG.



- The connection between electricity and magnetism was first discovered by Hans Christian Orsted in 1819.
- He demonstrated that a magnetic compass needle was affected by a current passing through a circuit.
- An electrical dipole is always surrounded by a corresponding magnetic field.
- The polarity of the field is determined by the direction of the current.

History

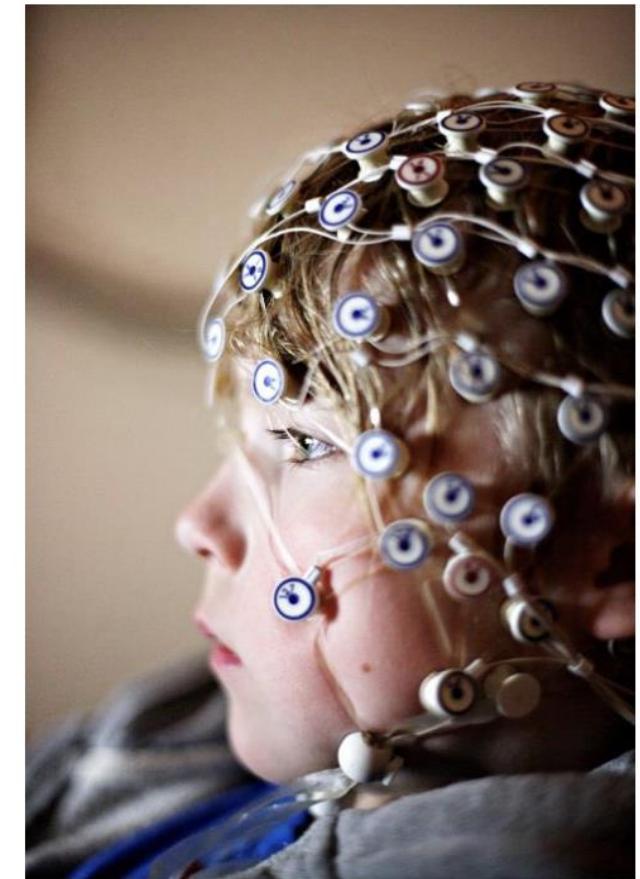


1929: **Hans Berger** developed the electroencephalography (=graphic representation of the difference in voltage between two different cerebral locations plotted over time) following the studies of Richard Caton in non-human animal species.

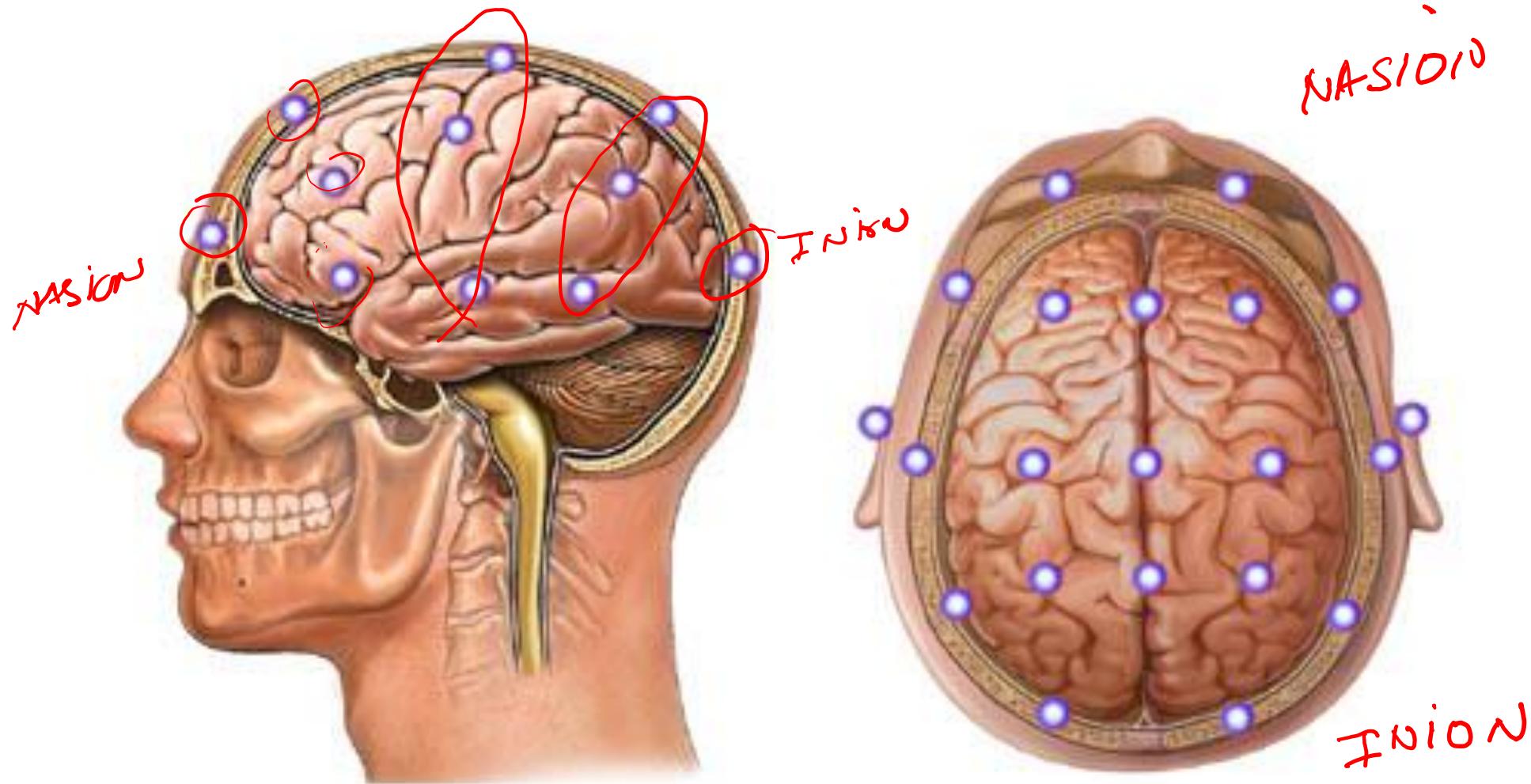
He described the human alpha and beta rhythms.

8-13 Hz
Berger waves

High density EEG recording



Electrode placing locations in EEG



Electrode location in EEG

The names of the electrode sites use **alphabetical abbreviations** that identify the lobe or area of the brain to which each electrode refers:

F = frontal

Fp = frontopolar

T = temporal

C = central

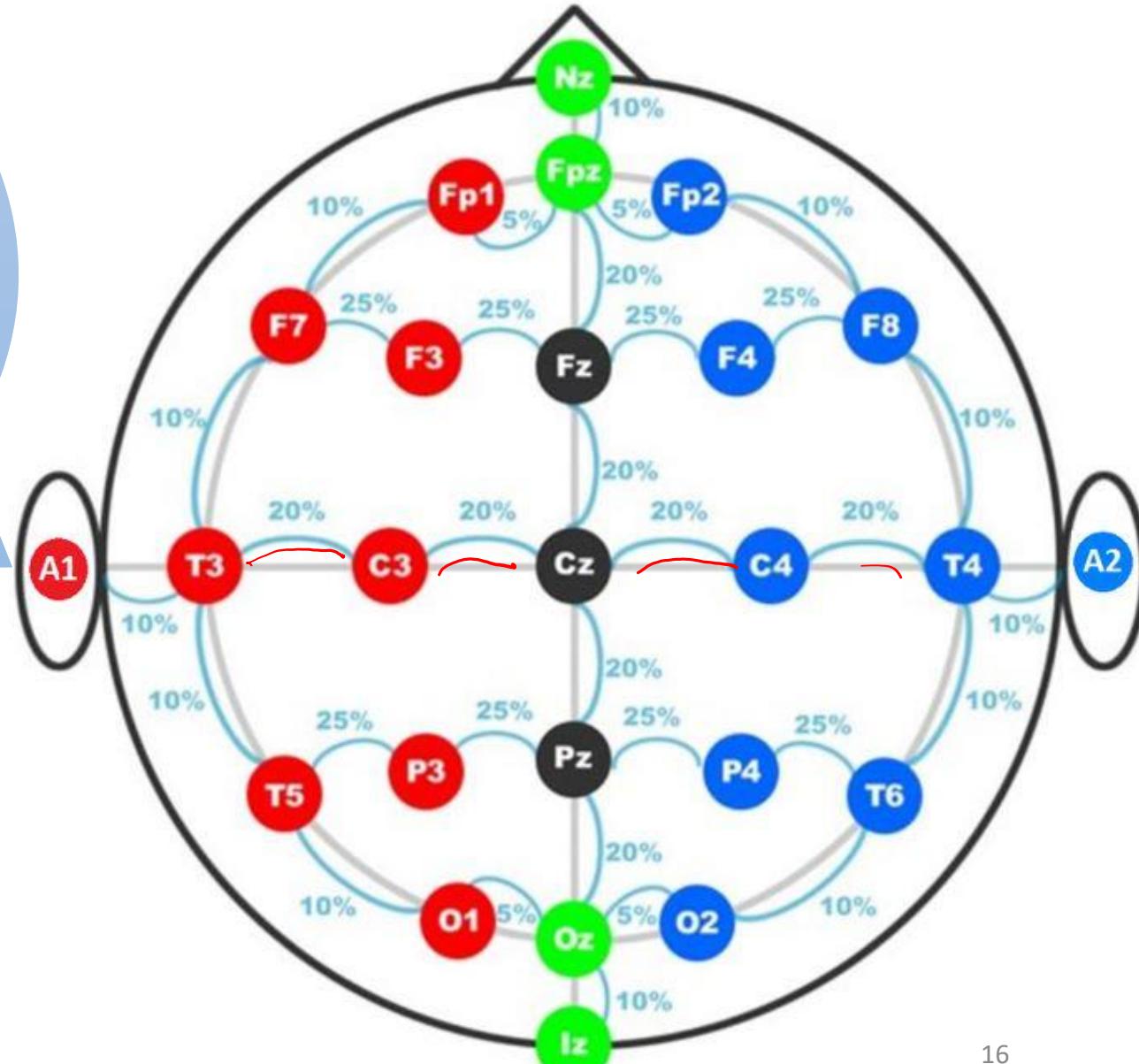
P = parietal

O = occipital

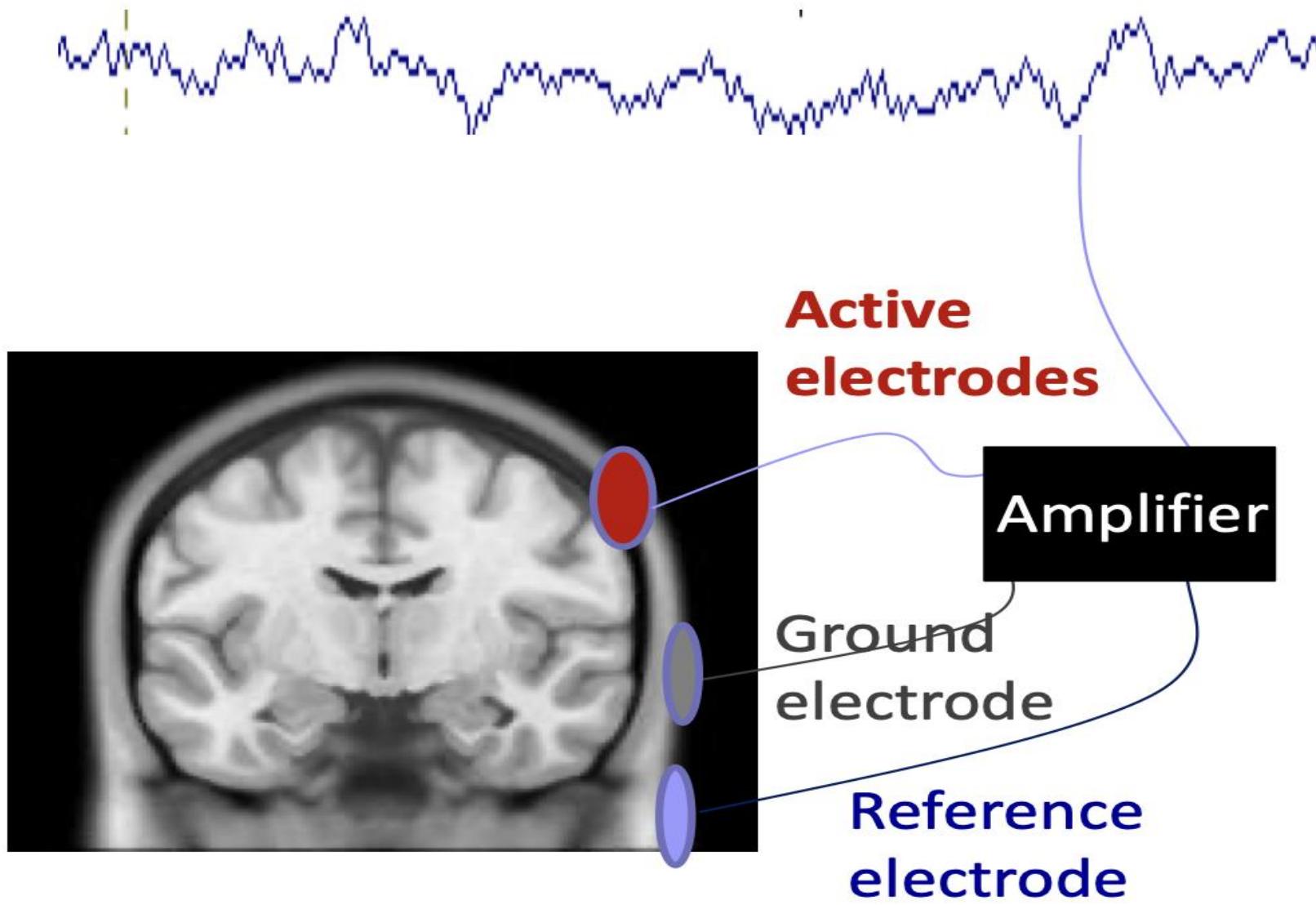
A = auricular (ear electrode)

“**z**” refers to an electrode placed on the mid-line

- ❖ **Even numbers** denote the **right side** of the head and **odd numbers** the left side of the head.



Measuring EEG...



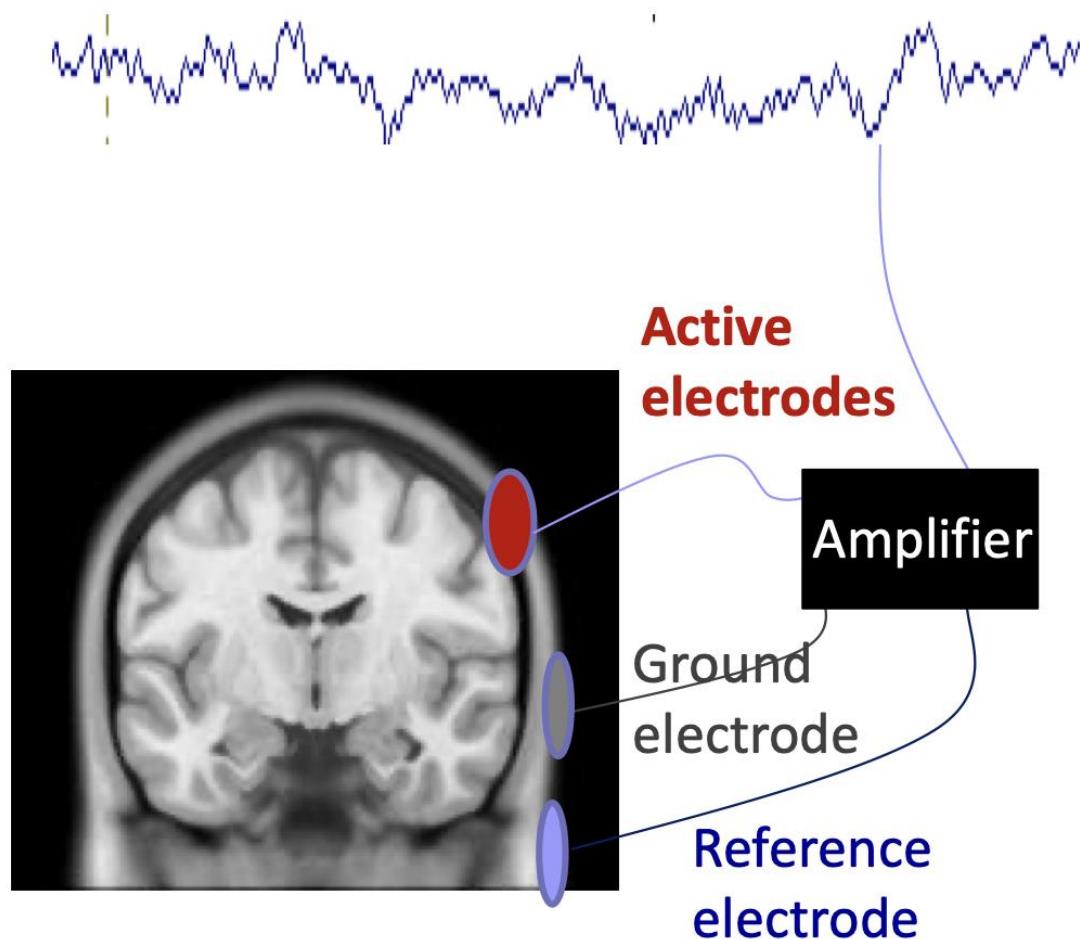
Measuring EEG...

Electric fields affecting measurements:

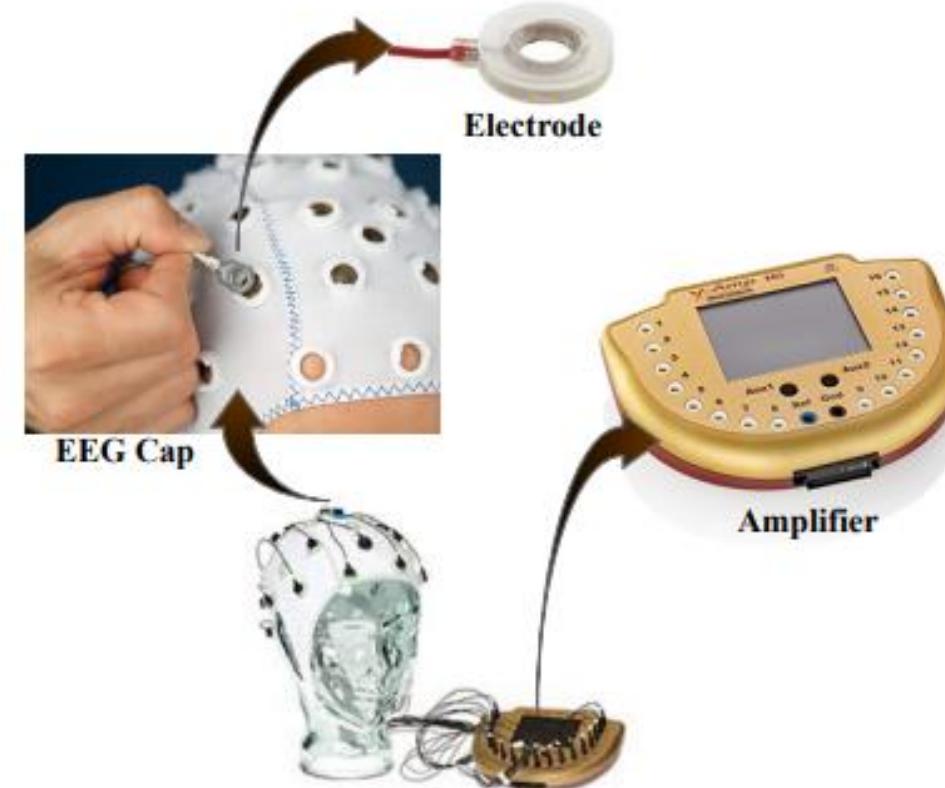
1. Static electric current
2. Electric noise
3. Brain activity

Ground = difference between the participant and the amplifier

- Subtracted from the active and reference electrode activity: A-G, R-G
- is the reference point in an **electrical circuit** from which voltages are measured, a common return path for **electric current**, or a direct physical connection to the **Earth**.



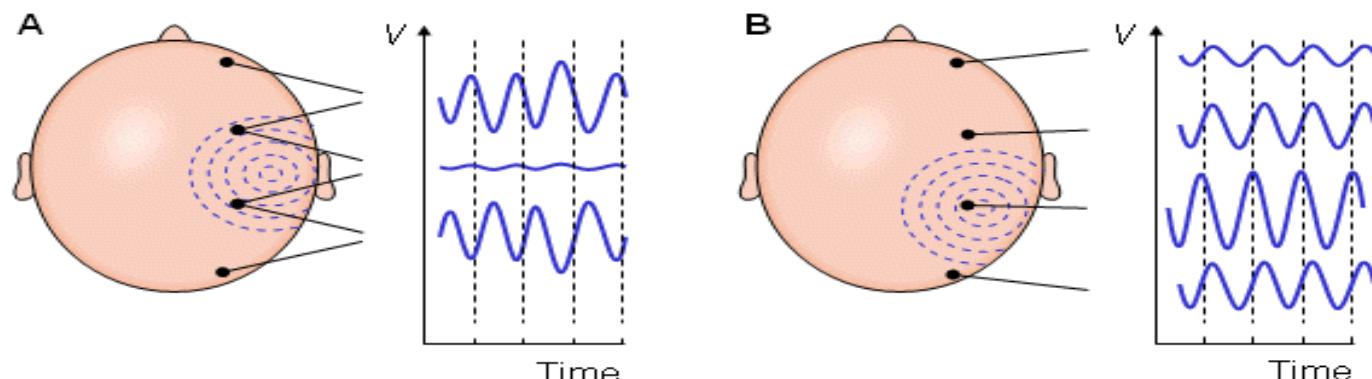
An EEG Device



Other Accessories

Measuring EEG...

- EEG records potential differences at the scalp using a set of electrodes and a reference.
- The ground electrode is important to eliminate noise from the amplifier circuit.
- The representation of the EEG channels is referred to as a montage
 - Unipolar/Referential \Rightarrow potential difference between electrode and designated reference
 - Bipolar \Rightarrow represents difference between adjacent electrodes (e.g. ECG, EOG)
- Potential differences are then amplified and filtered



Thank you!!



EEG Paradigms

Course Instructor

Dr. Annushree Bablani

Acknowledgments: Dr Sreeja SR

Direct (noninvasive) interfaces in EEG

- An event-related potential (ERP) is any measured brain response that is directly the result of a thought or perception. More formally, it is any stereotyped electrophysiological response to an internal or external stimulus.
- Direct Interfaces via EEG
 - VEP – Visual Evoked Potential
 - AEP – Auditory Evoked Potential
 - SSVEP – Steady-State Visual Evoked Potential
 - P300 – ERP elicited by infrequent, task-relevant stimuli.
 - ERS/ERD – Event related synchronization/desynchronization
 - SCP – Slow cortical potentials

Categorization of EEG based BCI paradigms

Evoked (Endogenous / Asynchronous)

- Subject must pay attention for a certain time to external cues (e.g. flashes, sounds, etc.)
- Cue-based

Spontaneous (Exogenous / Synchronous)

- No continuous attention to specific stimulus is necessary
- User-driven



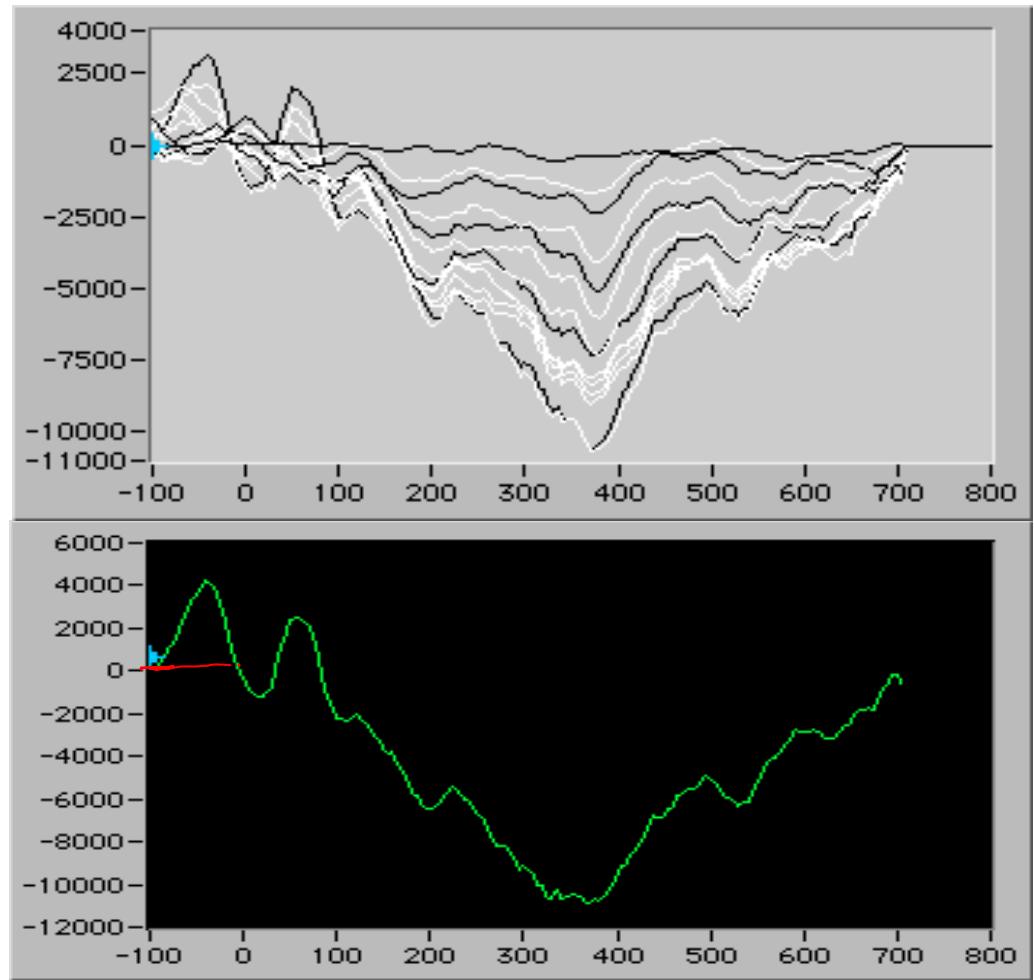
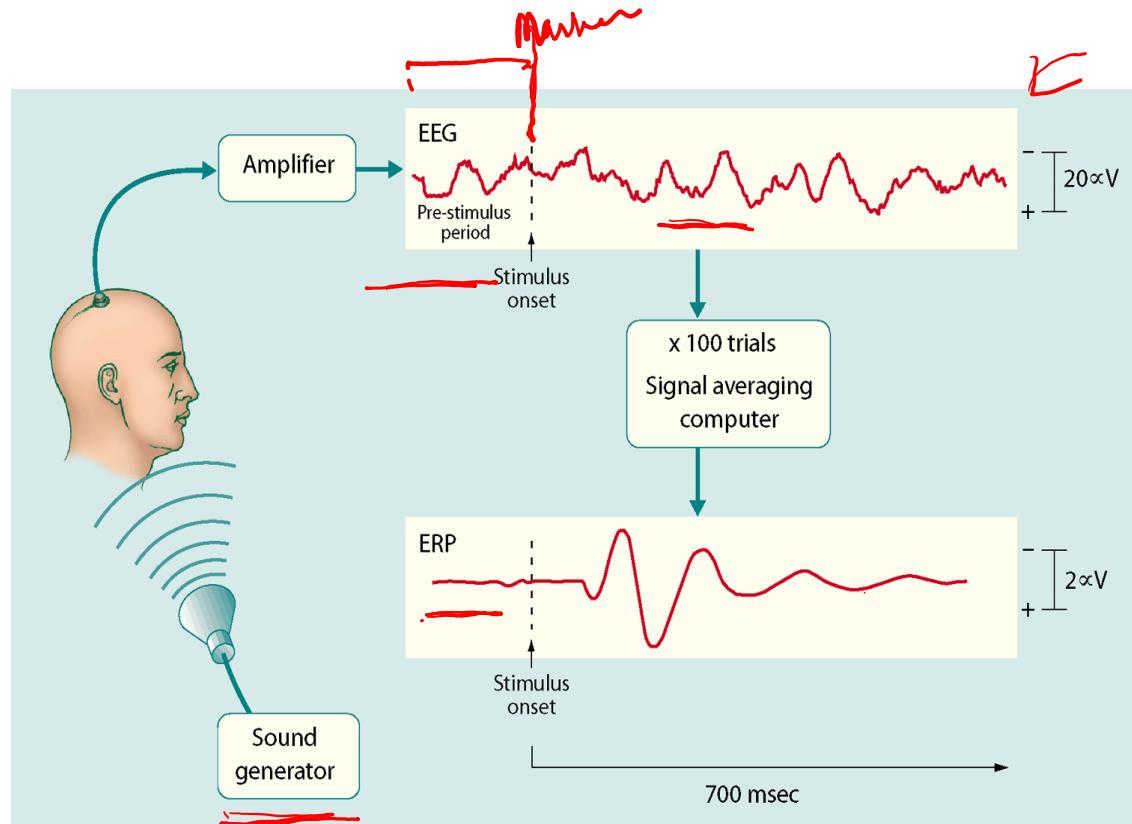
Categorization of EEG based BCI paradigms

ERD		Spontaneous
P300	→	Evoked
SSEP/AEP/VEP	→	Evoked
SCP	→	Spontaneous

Event Related Potentials (ERP)

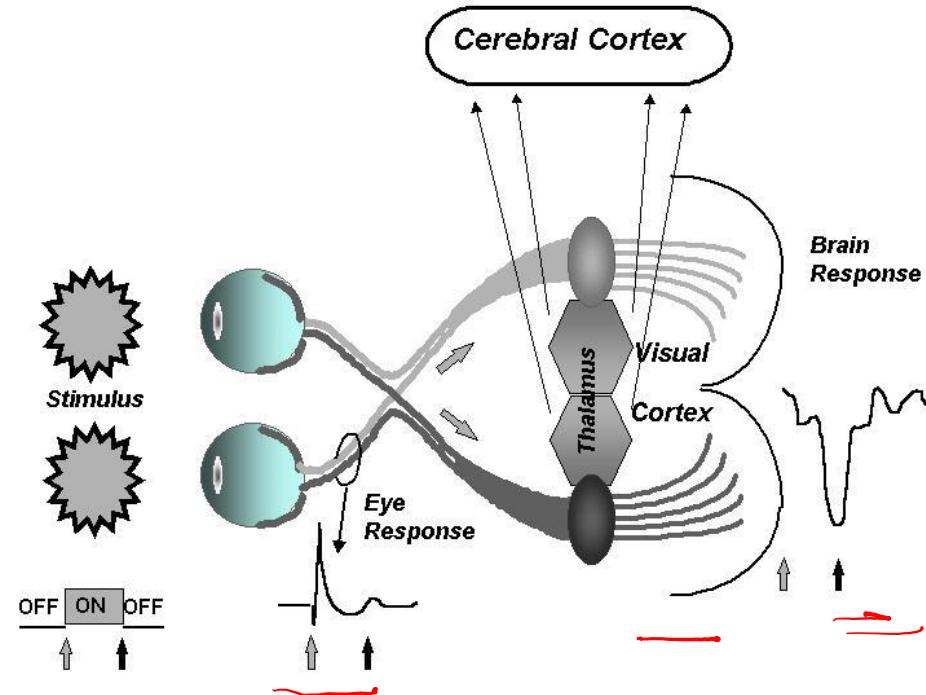
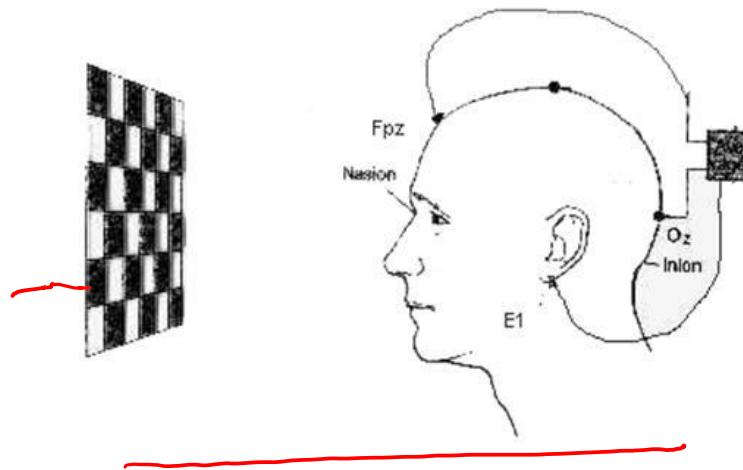
→ *reflections* → +ve P100
↓ -ve N100

- Averaging of trials following a stimulus



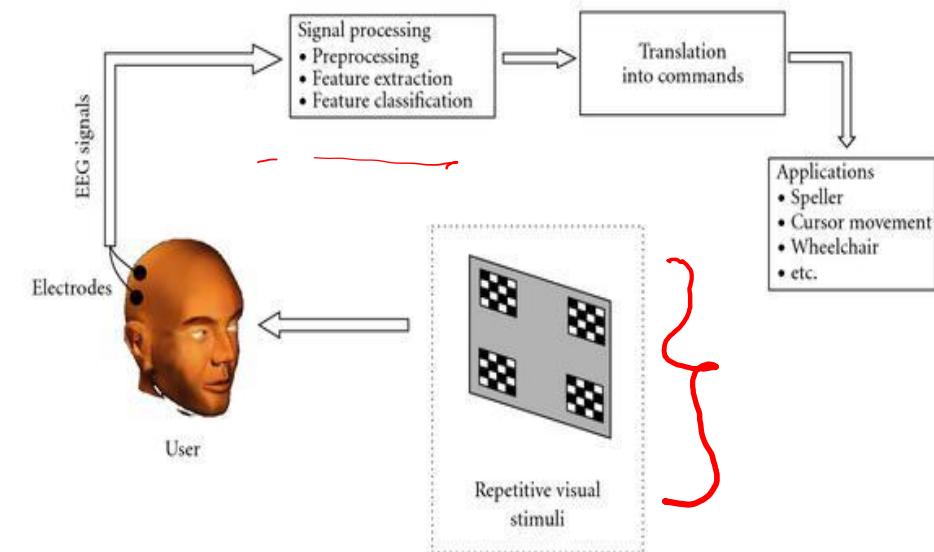
Visual Evoked Potential (VEP)

- Caused by Visual Stimulus
- Occurs with flashing lights (3-5 Hz)
- Have been used to monitor function during surgery for lesions involving the pituitary gland, optic nerve.
- Application:



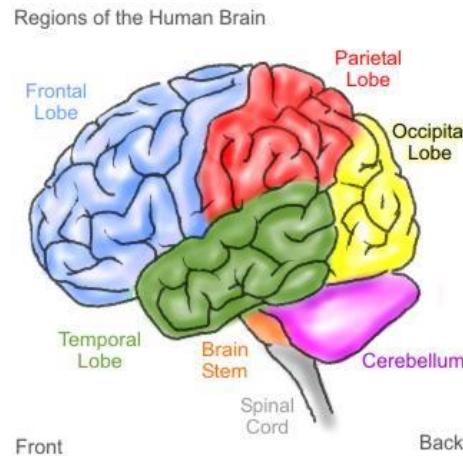
Steady-State Visual Evoked Potential (SSVEP)

- SSVEP are signals that are natural responses to visual stimulation at specific frequencies. When the retina is excited by a visual stimulus ranging from 3.5 Hz to 75 Hz, the brain generates electrical activity at the same (or multiples of) frequency of the visual stimulus.
- Excellent signal-to-noise ratio and relative immunity to artifacts.
- Applications:
 - SSVEP-controlled robots (Boston University)
 - User-friendly interface



P300 ← ER^I

- P300 is thought to reflect processes involved in stimulus evaluation or categorization.
- It is usually elicited using the oddball paradigm in which low-probability target items are inter-mixed with high-probability non-target (or "standard") items.
- Results in a positive curve on EEG after 300ms.
- Strongest signal at parietal lobe.

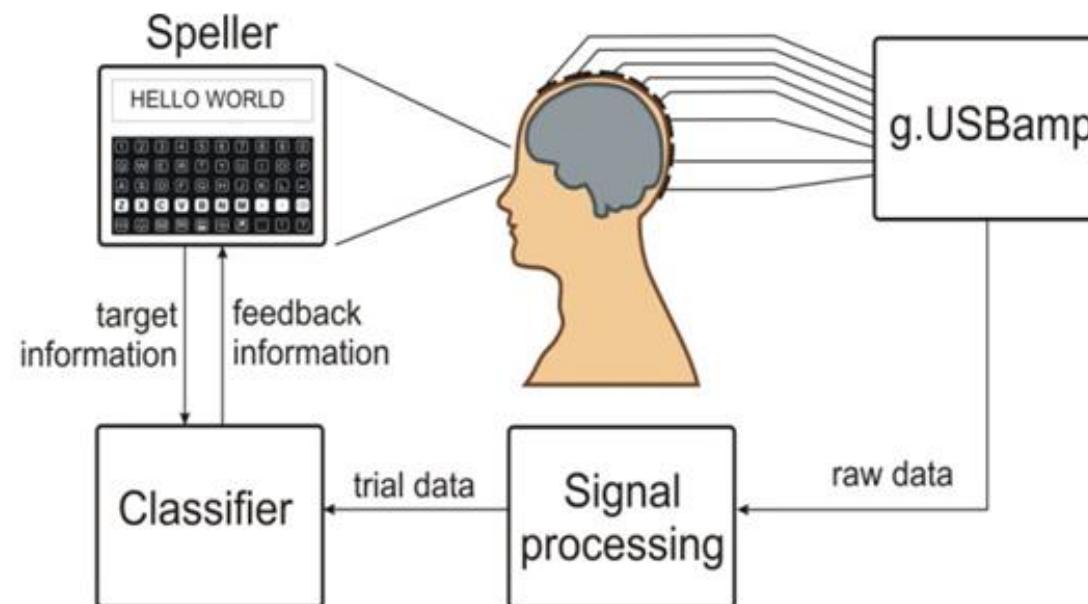


- 6x6 matrix of symbols
- Subject concentrates on a symbol (i.e. cell)
- Each row and column flashes twice
 - i.e., 2 target flashes vs. 10 non-target flashes
 - random order
 - for very short time (e.g. 100 ms)



P300

- (Farwell and Donchin 1988)
- 95% accuracy at 1 character per 26s

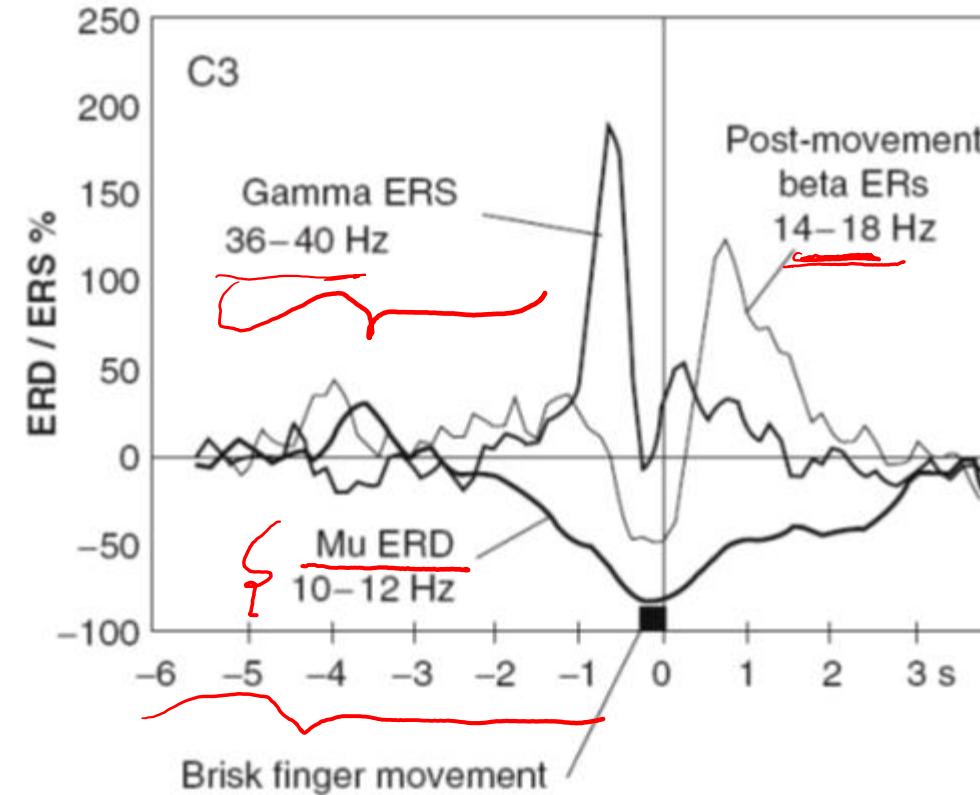


ERS/ERD

- Event-related desynchronization (ERD) and event-related synchronization (ERS) is the change of signal's power occurring in a given band relative to a reference interval.
- People have naturally occurred brain rhythms over areas of the brain concerned with touch and movement. When people imagine moving, these brain rhythms first become weaker, then stronger. These two changes are called ERD and ERS, respectively.
- **ERS**
 - oscillatory power increase
 - associated with activity decrease
- **ERD**
 - oscillatory power decrease
 - associated with activity increase

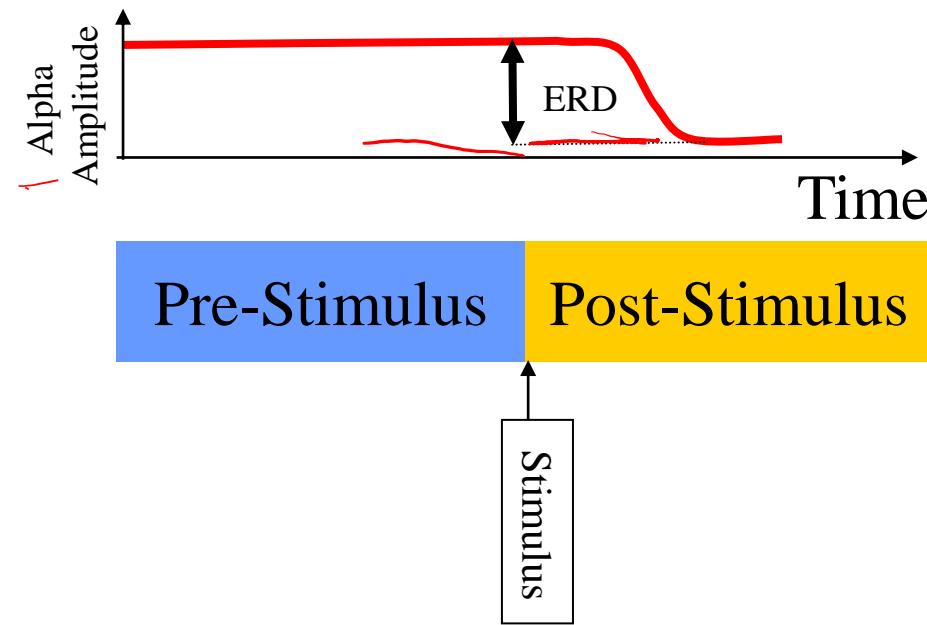
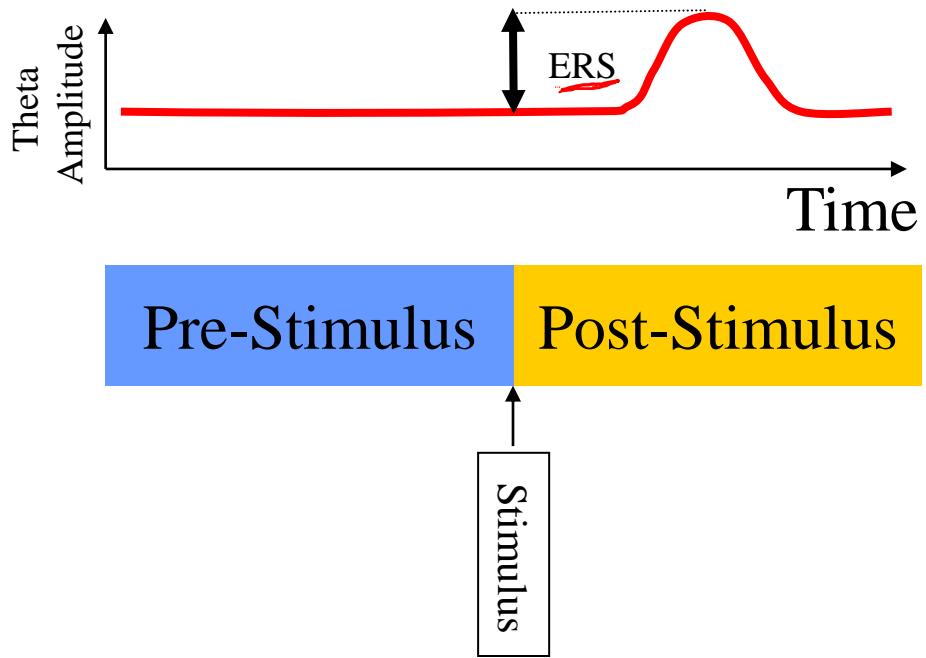
ERS/ERD

- The imagination of either a left or right hand movement results in:
 - An amplitude attenuation (Event-Related Desynchronization (ERD)) of μ (8-12Hz) and central beta EEG-rhythms (13-30Hz) at the contralateral sensorial motor representation area and,
 - in some cases, in an amplitude increase (Event-Related Synchronization (ERS)) within the γ -band (30-40Hz) at the ipsilateral hemisphere(6).

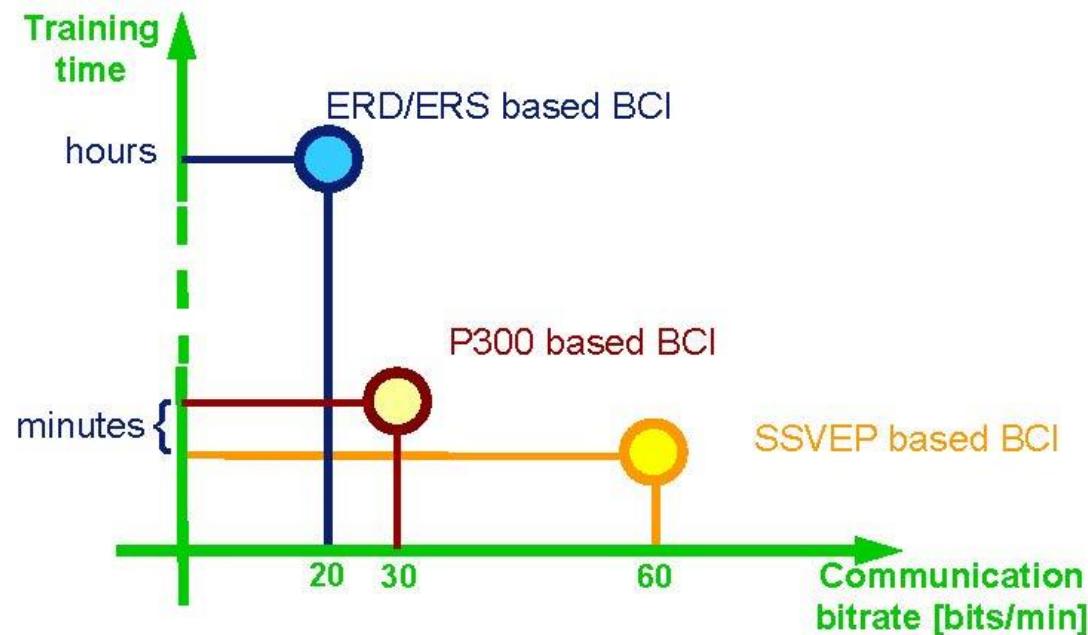


EEG recorded from C3 electrode.

4 8 13 * 2
8 13 * 2



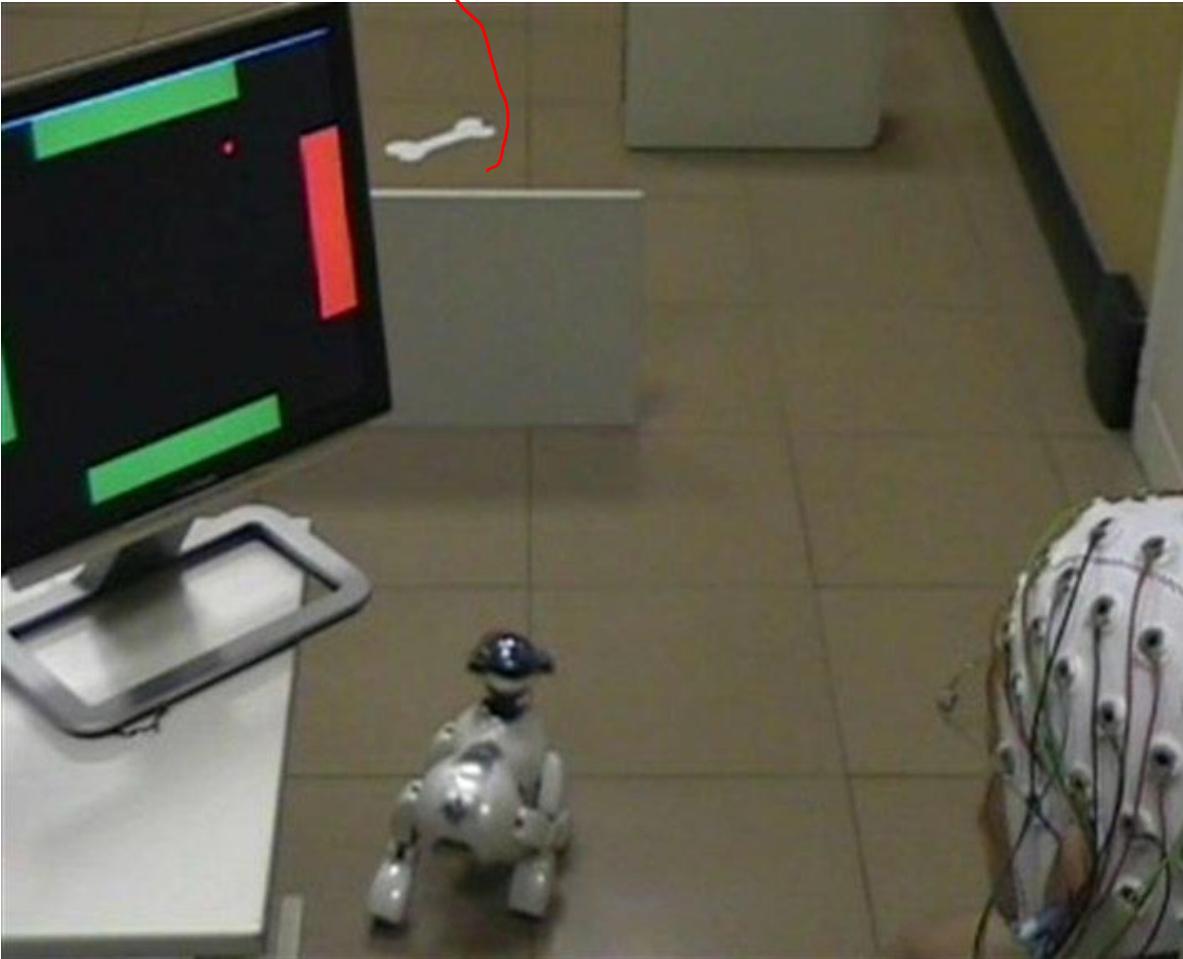
Communication Issues



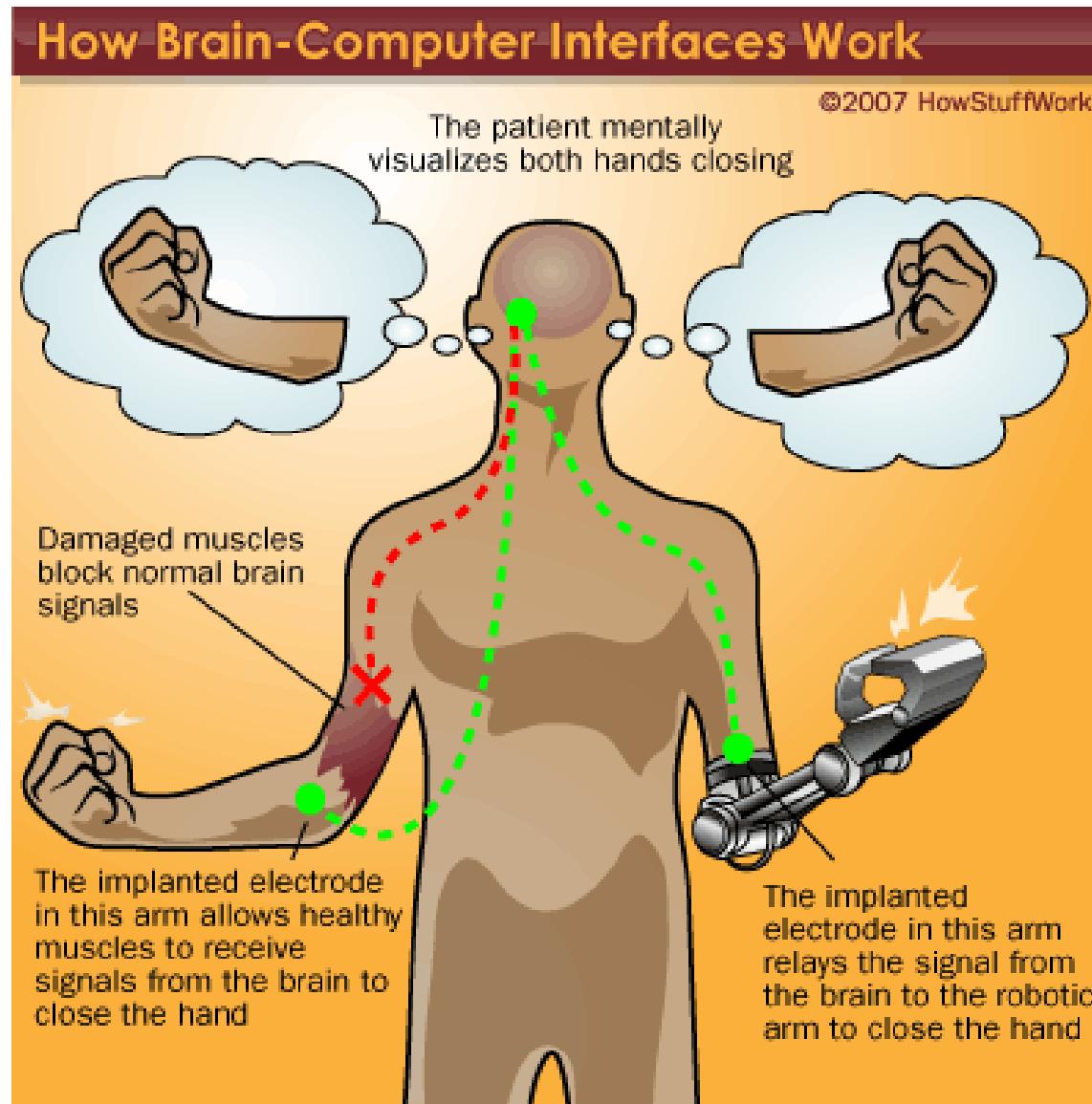
Typical training time versus communication bitrate for the three main types of noninvasive EEG based BCIs.

BCI Applications

BCI – operated robot



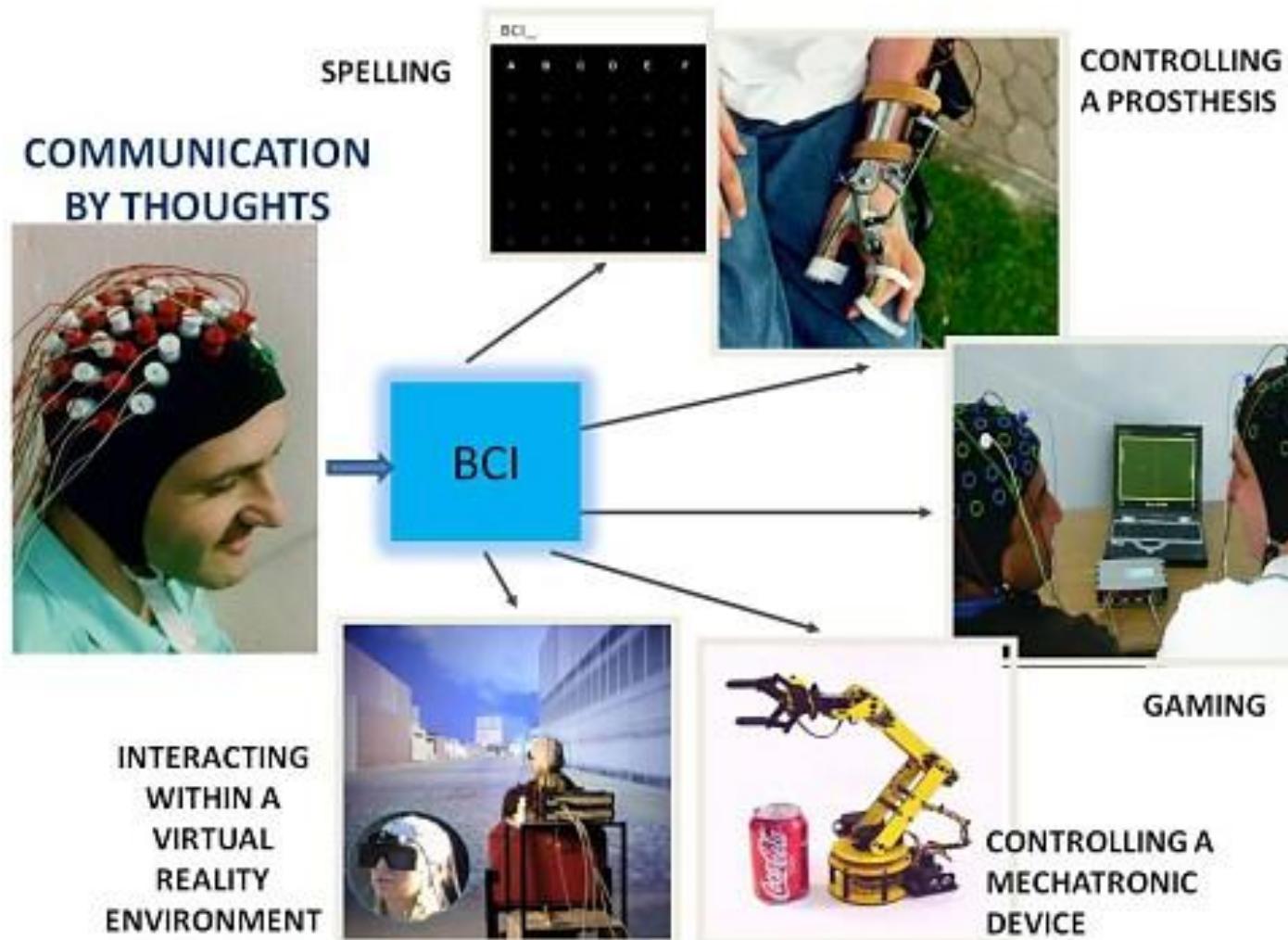
BCI Applications



BCI Applications



BCI Applications



Thank you!



EEG Variables

Course Instructor

Dr. Annushree Bablani

Acknowledgments: Dr. Sreeja S R

EEG

- Electroencephalogram (EEG) signals are useful for diagnosing various mental conditions such as epilepsy, memory impairments and sleep disorders.
- EEGs can indicate the general conscious state of a person, e.g., asleep, awake, anaesthetized, since each state is correlated with particular EEG patterns.
- A flat EEG (no electrical activity) is clinical evidence of death.

Why EEG?

- Hardware costs are significantly lower than those of most other techniques.
- EEG sensors can be used in more places than fMRI, SPECT, PET, MRS, or MEG, as these techniques require bulky and immobile equipment.
- EEG has very high temporal resolution, on the order of milliseconds rather than seconds, commonly recorded at sampling rates between 250 and 2000 Hz thus a valuable tool for research and diagnosis.
- EEG is relatively tolerant of subject movement, unlike most other neuro imaging techniques. There even exist methods for minimizing, and even eliminating movement artifacts in EEG data.
- EEG is silent, which allows for better study of the responses to auditory stimuli.
- EEG does not involve exposure to high-intensity (>1 Tesla) magnetic fields, as in some of the other techniques, especially MRI and MRS. These can cause a variety of undesirable issues with the data, and also prohibit use of these techniques with participants that have metal implants in their body, such as metal-containing pacemaker.
- EEG can be used in subjects who are incapable of making a motor response.
- EEG is a powerful tool for tracking brain changes during different phases of life. EEG sleep analysis can indicate significant aspects of the timing of brain development, including evaluating adolescent brain maturation.

EEG Disadvantages

- Low spatial resolution on the scalp. fMRI, for example, can directly display areas of the brain that are active, while EEG requires intense interpretation just to hypothesize what areas are activated by a particular response.
- EEG poorly determines neural activity that occurs below the upper layers of the brain (the cortex).
- Unlike PET and MRS, cannot identify specific locations in the brain at which various neurotransmitters, drugs, etc. can be found.
- Often takes a long time to connect a subject to EEG, as it requires precise placement of dozens of electrodes around the head and the use of various gels, saline solutions, and/or pastes to keep them in place. Whereas a general rule it takes considerably less time to prepare a subject for MEG, fMRI, MRS, and PET.

EEG Pioneers

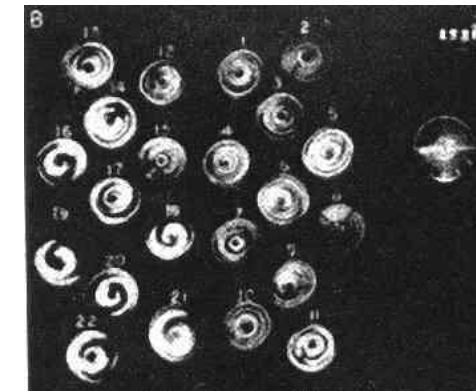
In 1929, Hans Berger

- Recorded brain activity from the closed skull
- Reported brain activity changes according to the functional state of the brain
 - Sleep
 - Hypnosis
 - Pathological states (epilepsy)



In 1957, Gray Walter

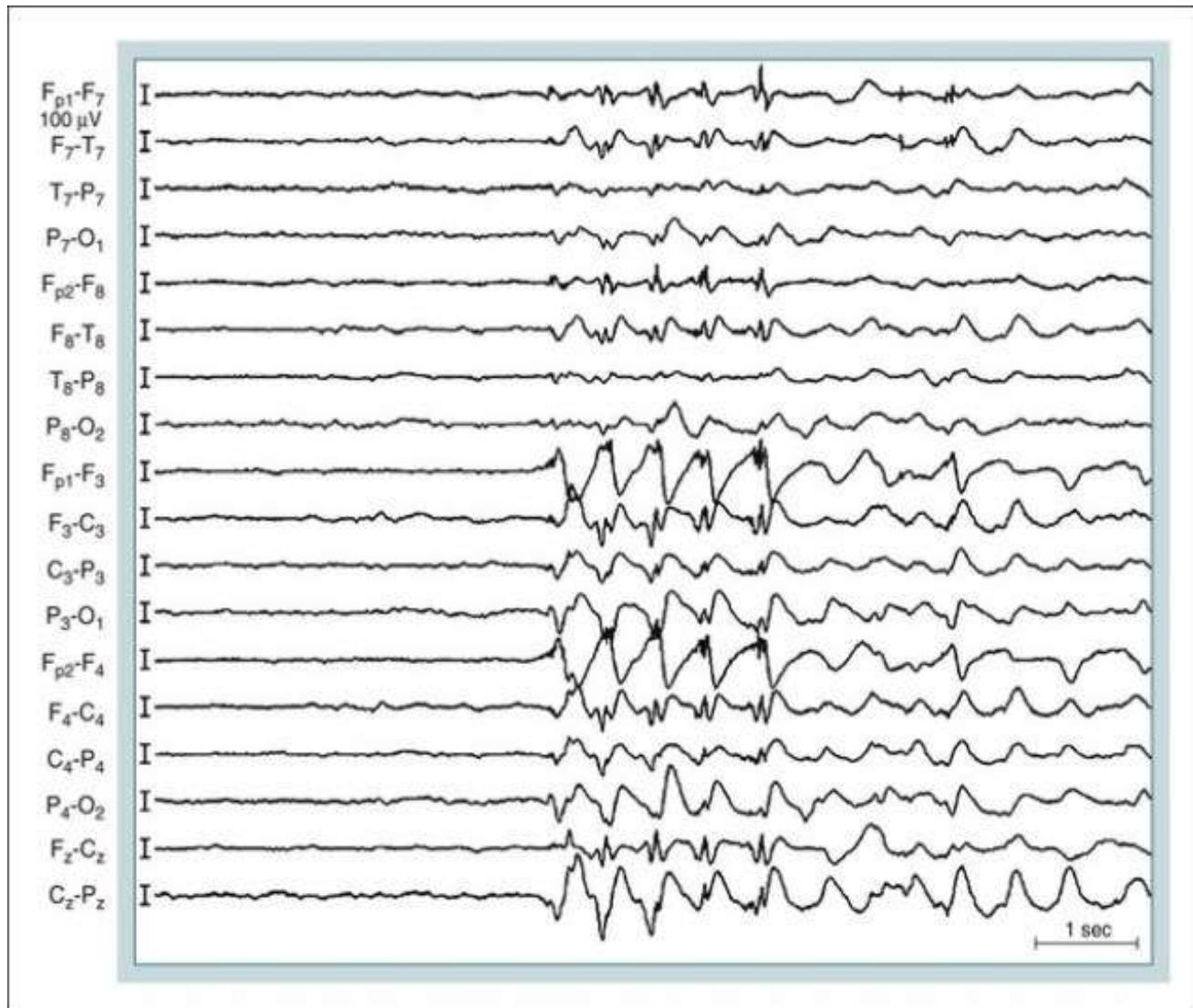
- Makes recordings with large numbers of electrodes
- Visualizes brain activity with the toposcope
- Shows that brain rhythms change according to the mental task demanded



Representation of EEG channels:

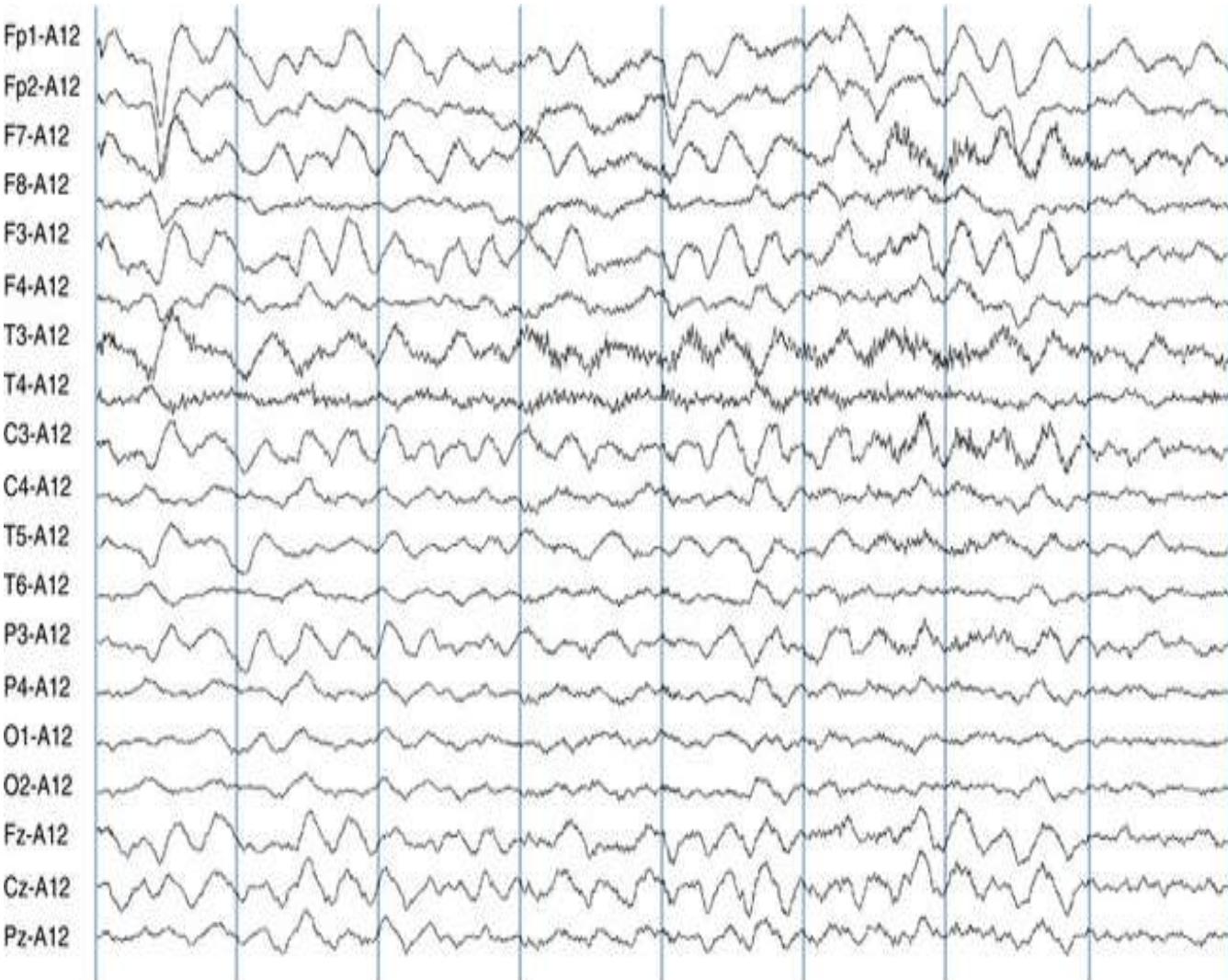
The representation of the EEG channels (i.e., waveform) is referred to as a **montage**.

- **Sequential montage:** Each channel represents the difference between two adjacent electrodes. The entire montage consists of a series of these channels. For example, the channel "Fp1-F3" represents the difference in voltage between the Fp1 electrode and the F3 electrode. The next channel in the montage, "F3-C3," represents the voltage difference between F3 and C3, and so on through the entire array of electrodes.



Representation of EEG channels:

- **Referential montage:** Each channel represents the difference between a certain electrode and a designated reference electrode. There is no standard position for this reference; it is, however, at a different position than the "recording" electrodes. Midline positions are often used because they do not amplify the signal in one hemisphere vs. the other. Another popular reference is "linked ears," which is a physical or mathematical average of electrodes attached to both earlobes or mastoids.

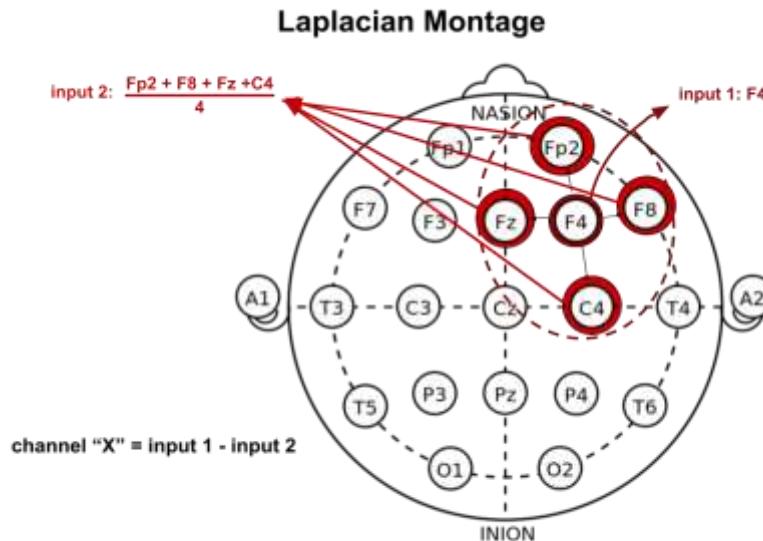
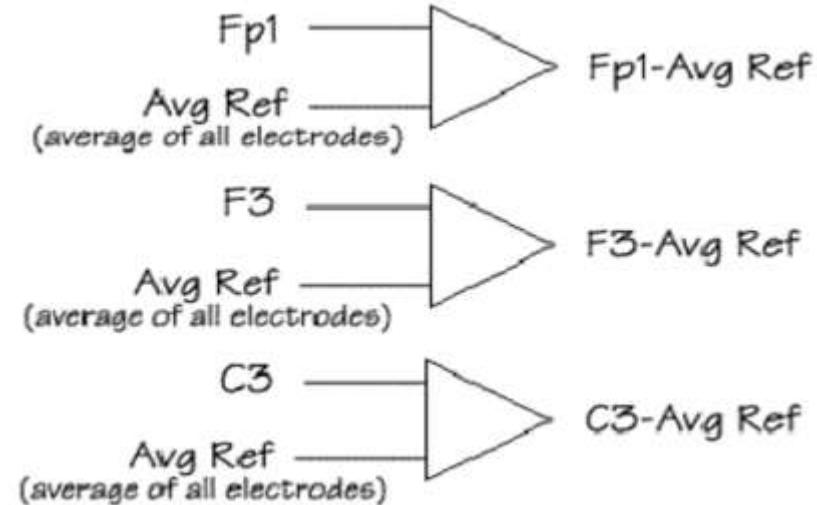


Representation of EEG channels:

- **Average reference montage:**

The outputs of all of the amplifiers are summed and averaged, and this averaged signal is used as the common reference for each channel.

- **Laplacian montage:** Each channel represents the difference between an electrode and a weighted average of the surrounding electrodes.



EEG Rhythms

- Generally grouped by frequency: (amplitudes are about 100 μ V max)

Type	Frequency	Location	Use
Delta	<4 Hz	everywhere	occur during sleep, coma
Theta	4-7 Hz	temporal and parietal	correlated with emotional stress (frustration & disappointment)
Alpha	8-15 Hz	occipital and parietal	reduce amplitude with sensory stimulation or mental imagery
Beta	16-30 Hz	parietal and frontal	can increase amplitude during intense mental activity
Gamma	>30 Hz	Somatosensory cortex	A decrease in gamma-band activity is associated with cognitive decline
Mu	8-12 Hz	frontal (motor cortex)	diminishes with movement or intention of movement
Lambda	sharp, jagged	occipital	correlated with visual attention
Vertex			higher incidence in patients with epilepsy or encephalopathy

Alpha Rhythm

Frequency: 8 – 15 Hz

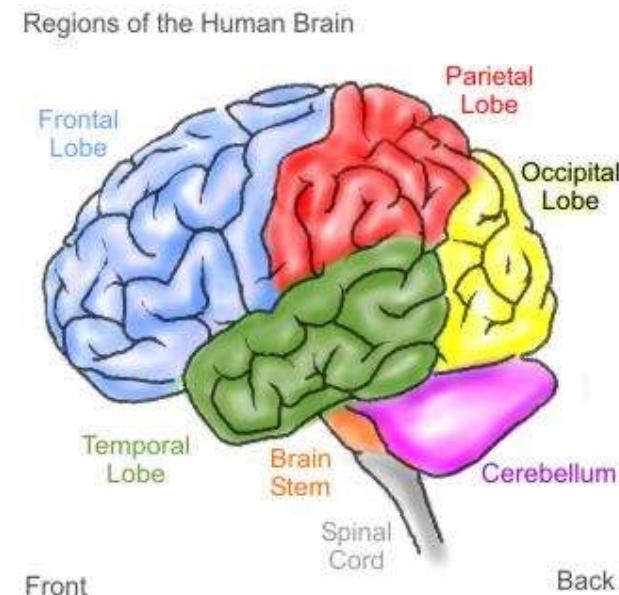
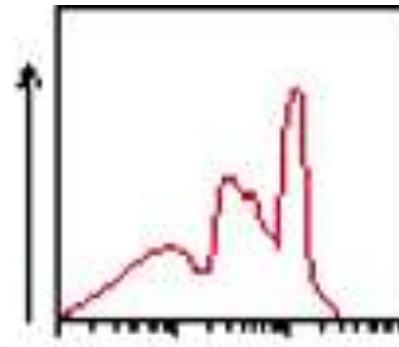
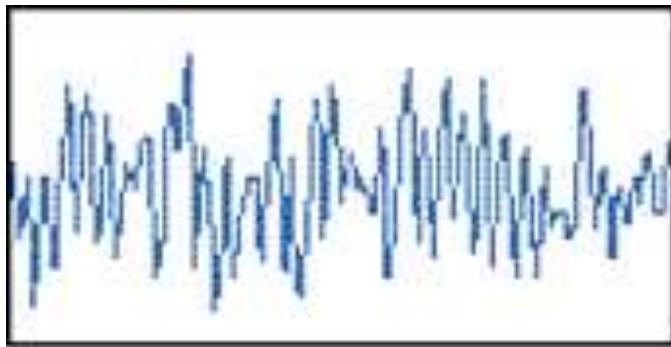
Amplitude: 5 – 100 microVolt

Location: Occipital, Parietal

State of Mind: Alert Restfulness

Source: Oscillating thalamic pacemaker neurons

Alpha blockade occurs when new stimulus is processed



Beta Rhythm

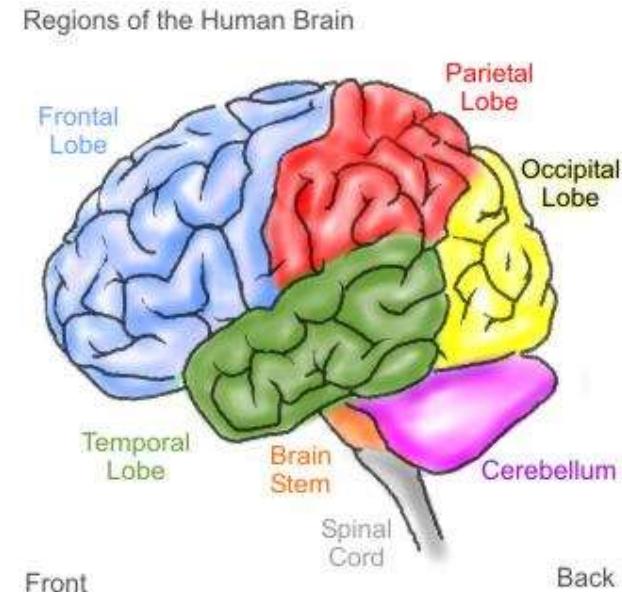
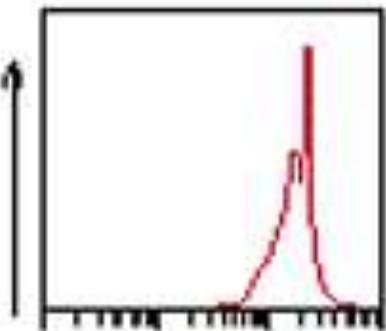
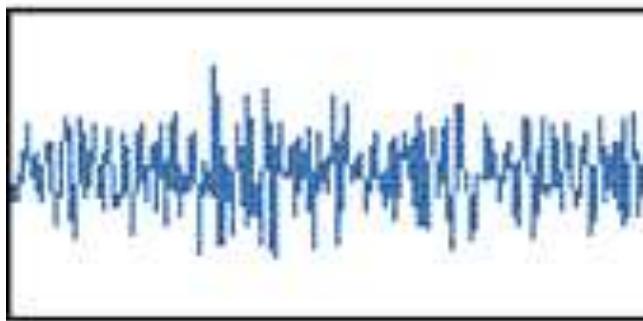
Frequency: 16 – 30 Hz

Amplitude: 2 – 20 microVolt

Location: Frontal

State of Mind: Mental Activity

Reflects specific information processing between cortex and thalamus



Delta Rhythm

Frequency: 1 – 4 Hz

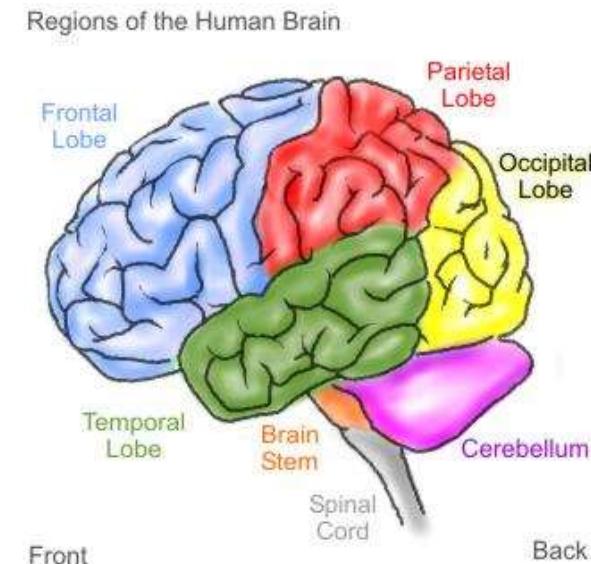
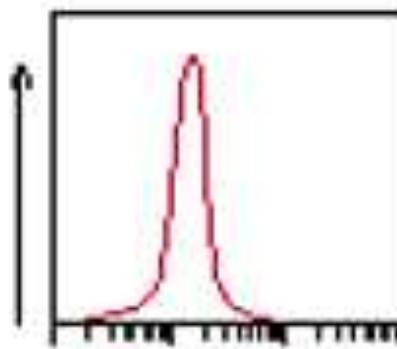
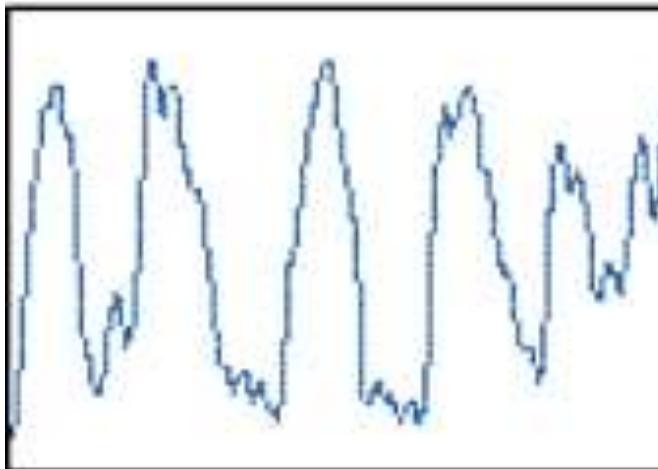
Amplitude: 20 – 200 microVolt

Location: Variable

State of Mind: Deep sleep

Oscillations in Thalamus and deep cortical layers

Usually inhibited by ARAS (Ascending Reticular Activation System)



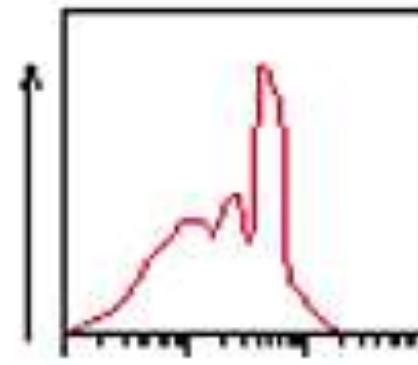
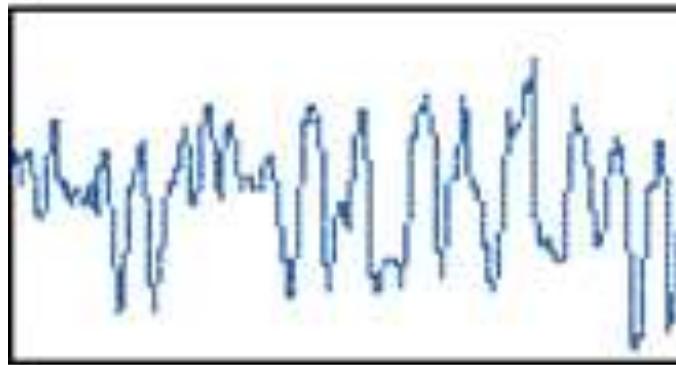
Theta Rhythm

Frequency: 4 – 7 Hz

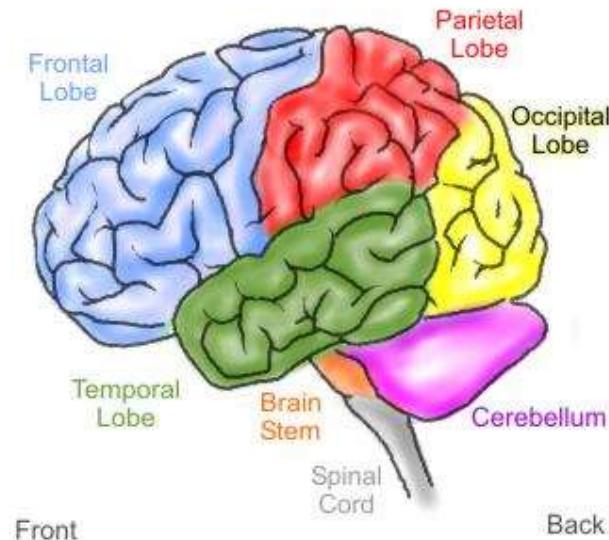
Amplitude: 5 – 100 microVolt

Location: Frontal, Temporal

State of Mind: Sleepiness



Regions of the Human Brain



Mu Waves

- Studied since 1930s
- Found in Motor Cortex
- Amplitude suppressed by Physical Movements, or *intent to move physically*
- (Wolpaw, et al 1991) trained subjects to control the mu rhythm by visualizing motor tasks to move a cursor up and down (1D)
- (Wolpaw and McFarland 2004) used a linear combination of Mu and Beta waves to control a 2D cursor.
- Weights were learned from the users in real time.
- Cursor moved every 50ms (20 Hz)
- 92% “hit rate” in average 1.9 sec

Alpha and Beta Waves

- Studied since 1920s
- Found in Parietal and Frontal Cortex
- Relaxed - Alpha has high amplitude
- Excited - Beta has high amplitude
- So, Relaxed -> Excited
means Alpha -> Beta

Variables used in EEG measurement:

Frequency:

- Frequency refers to rhythmic repetitive activity (in Hz). The frequency of EEG activity can have different properties including:
- **Rhythmic.** EEG activity consisting in waves of approximately constant frequency.
- **Arrhythmic.** EEG activity in which no stable rhythms are present.
- **Dysrhythmic.** Rhythms and/or patterns of EEG activity that characteristically appear in patient groups or rarely seen in healthy subjects.

Variables used in EEG measurement:

Voltage: Voltage refers to the average voltage or peak voltage of EEG activity.

- **Attenuation** (synonyms: suppression, depression). Reduction of amplitude of EEG activity resulting from decreased voltage.
- **Hypersynchrony.** Seen as an increase in voltage and regularity of rhythmic activity, or within the alpha, beta, or theta range. The term implies an increase in the number of neural elements contributing to the rhythm.
- **Paroxysmal.** Activity that reaching (usually) quite high voltage and ending with an abrupt return to lower voltage activity. Though the term does not directly imply abnormality, much abnormal activity is paroxysmal.

Variables used in EEG measurement:

Morphology: Morphology refers to the shape of the waveform. The shape of a wave or an EEG pattern is determined by the frequencies that combine to make up the waveform and by their phase and voltage relationships. Wave patterns can be described as being:

- **Monomorphic.** Distinct EEG activity appearing to be composed of one dominant activity
- **Polymorphic.** Distinct EEG activity composed of multiple frequencies that combine to form a complex waveform.
- **Sinusoidal.** Waves resembling sine waves. Monomorphic activity usually is sinusoidal.
- **Transient.** An isolated wave or pattern that is distinctly different from background activity.
 - **Spike:** a transient with a pointed peak and duration from 20 to less than 70 msec.
 - **Sharp wave:** a transient with a pointed peak and duration of 70-200 msec.

Variables used in EEG measurement:

Synchrony:

- Synchrony refers to the simultaneous appearance of rhythmic or morphologically distinct patterns over different regions of the head, either on the same side (unilateral) or both sides (bilateral).

Periodicity:

- Periodicity refers to the distribution of patterns or elements in time (e.g., the appearance of a particular EEG activity at more or less regular intervals). The activity may be generalized, focal or lateralized.

Thank You!



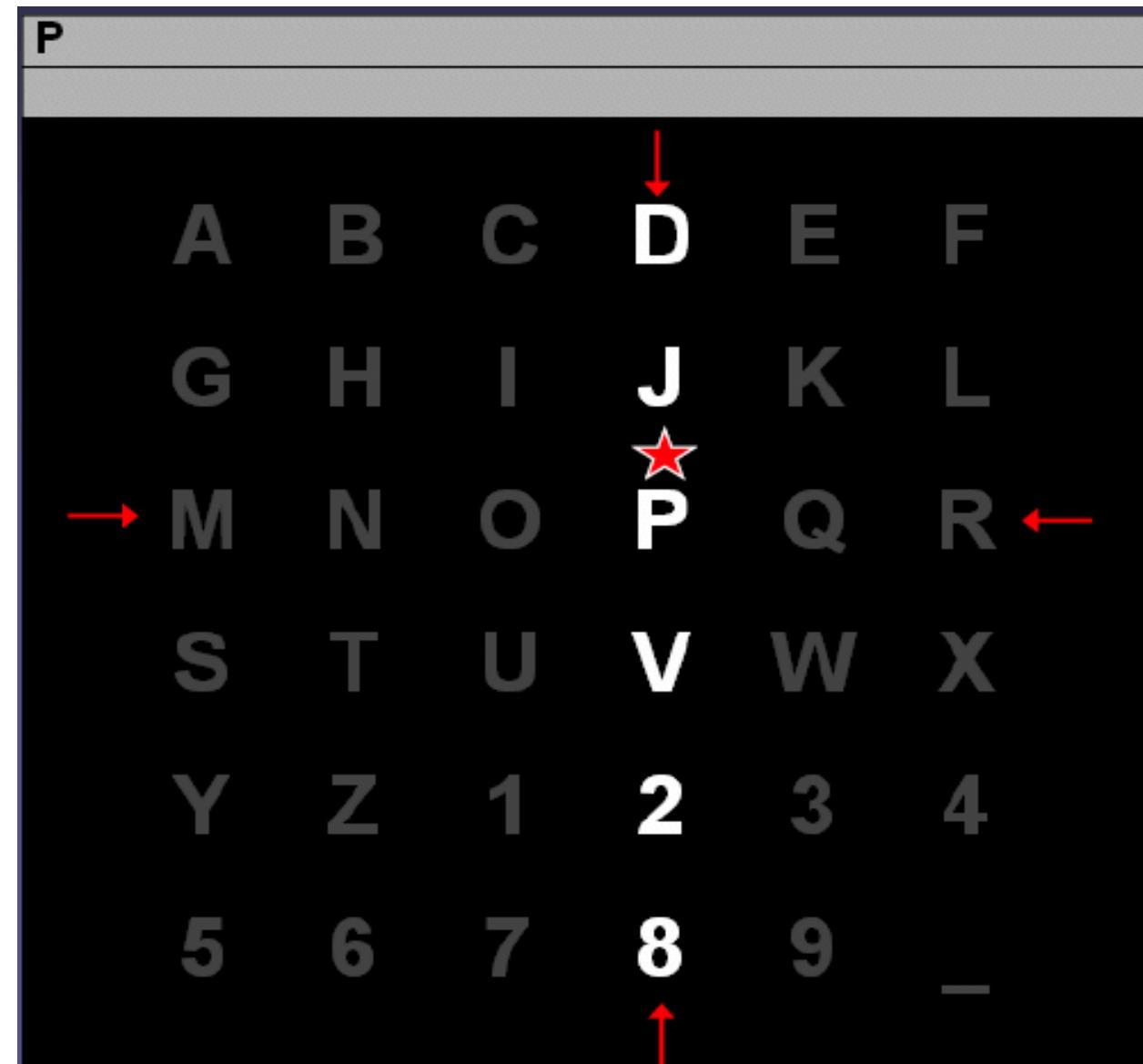
EEG Artifacts

Course Instructors

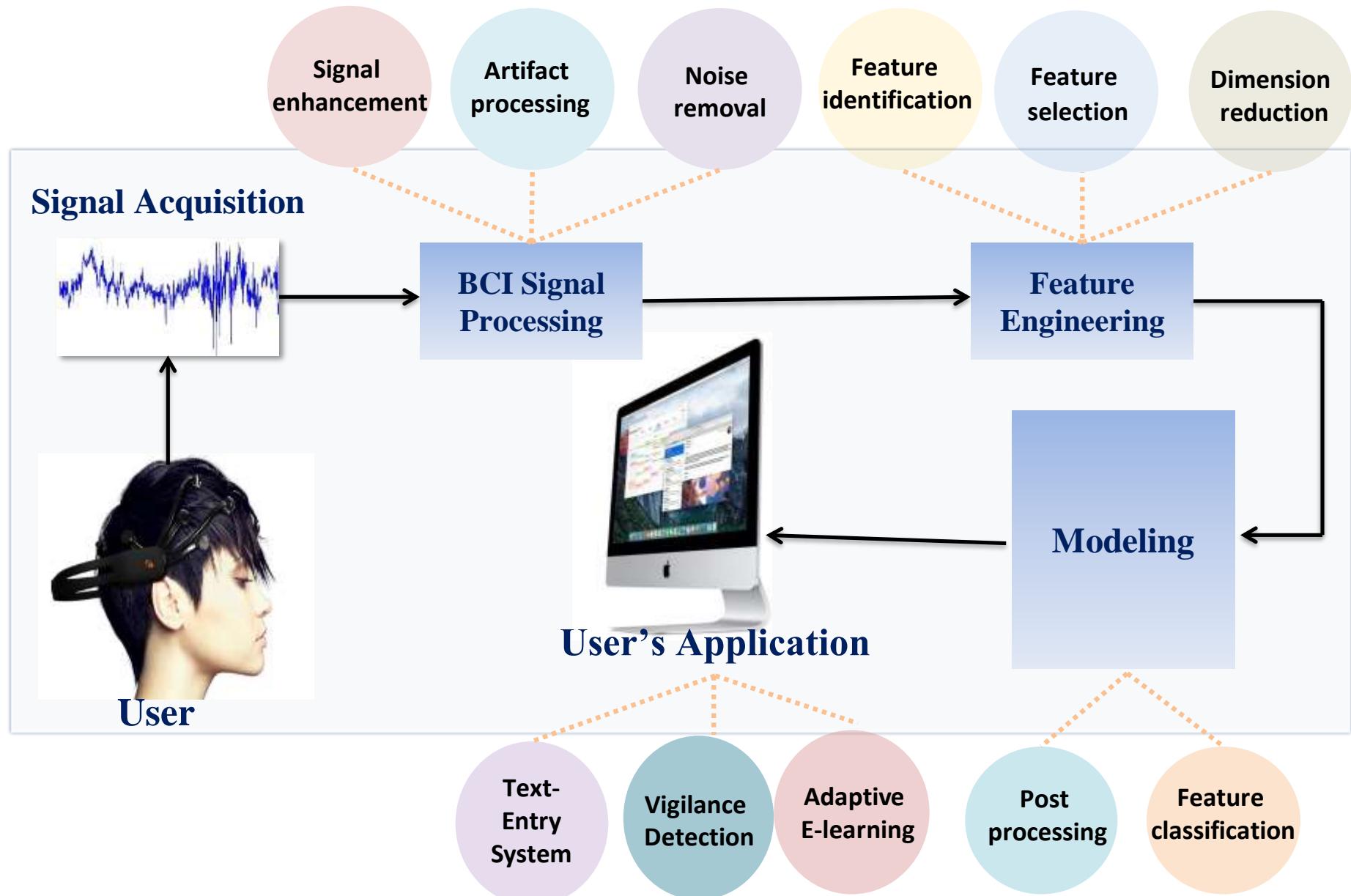
Dr. Annushree Bablani

Acknowledgments: Dr. Sreeja S R

EEG Paradigms



EEG based BCI System Development



Recording the EEG

- **EEG electrodes:**
 - Small metal discs usually made of stainless steel, tin, gold or silver covered with a silver chloride coating.
 - They are placed on the scalp in spatial positions using the International 10/20 system.



Fig: EEG cables showing the disc electrodes to which electrode gel is applied and applied to the subject's scalp.

Recording the EEG

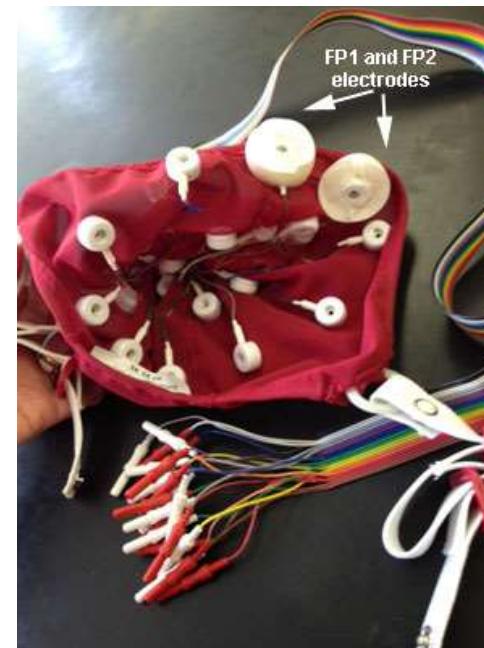
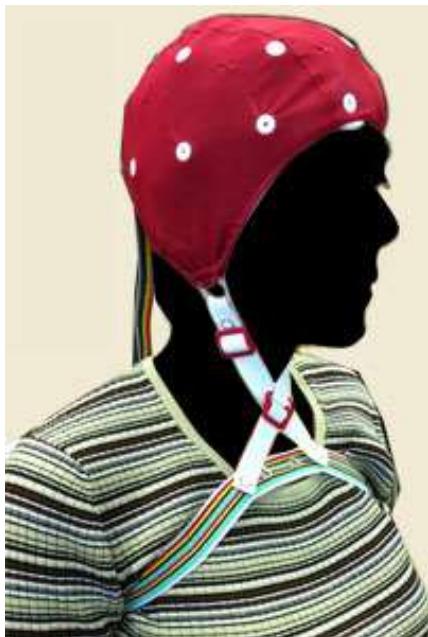


Fig: Many recording systems use a cap into which electrodes are embedded; this facilitates recordings when high density arrays of electrodes are needed or when comparing recording sites. The image to the right shows the inside of such a cap.

Recording the EEG

- **Electrode gel:**

- It acts as a malleable extension of the electrode, so that the movement of the electrodes cables is less likely to produce artifacts.
- The gel maximizes skin contact and allows for a low-resistance recording through the skin.

- **Impedance**

- A measure of the impediment to the flow of alternating current, measured in ohms at a given frequency.
- Larger numbers mean higher resistance to current flow.
- The higher the impedance of the electrode, the smaller the amplitude of the EEG signal.
- In EEG studies, should be at least 100 ohms or less and no more than 5 kohm.

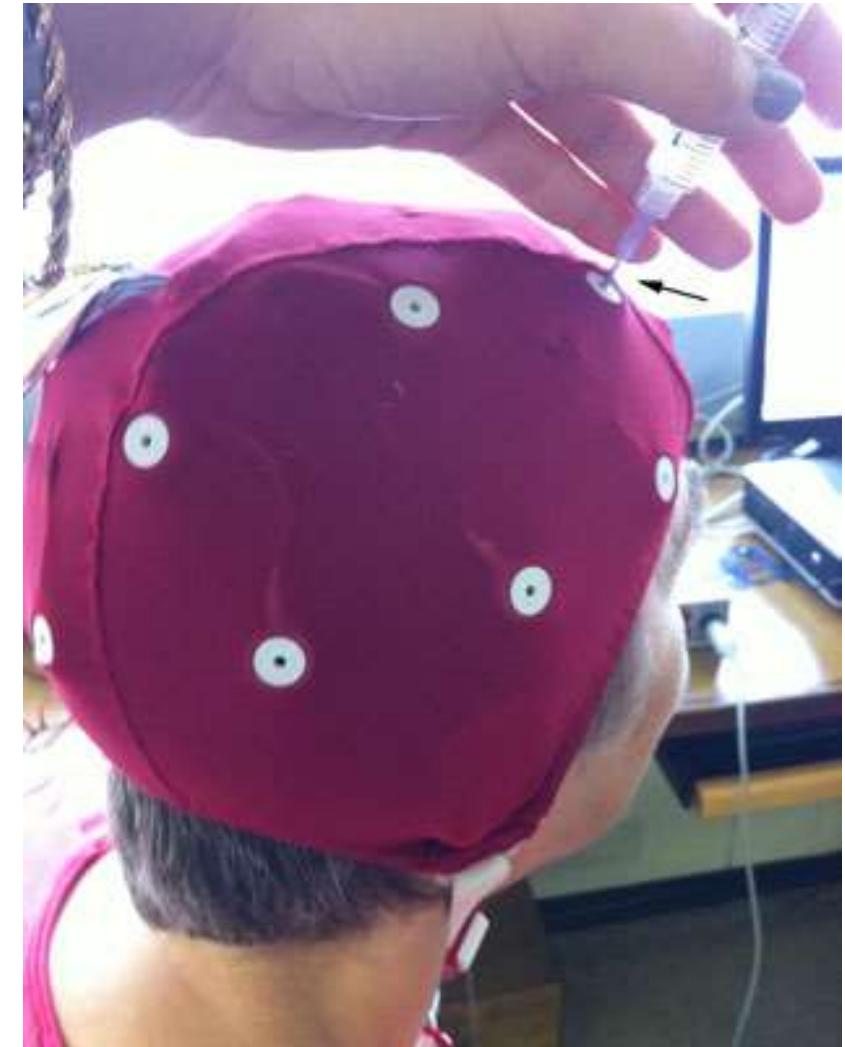


Fig: The electrolytic gel is injected into each cavity until a small amount comes out the hole in the mount. With a moderate amount of downward pressure, the syringe with a blunt needle is rapidly rocked back and forth.

EEG Artifacts

- The electrical artifacts that is not of cerebral origin.
- Anything that is NOT of cerebral origin is termed as **ARTIFACT**
- Physiological and Electrophysiological artifacts.
- Physiological – source (generated other than brain ie. Body)
- Electrophysiological – arise outside the body - equipment and environment
- Some readily distinguished, others closely resemble cerebral activity.

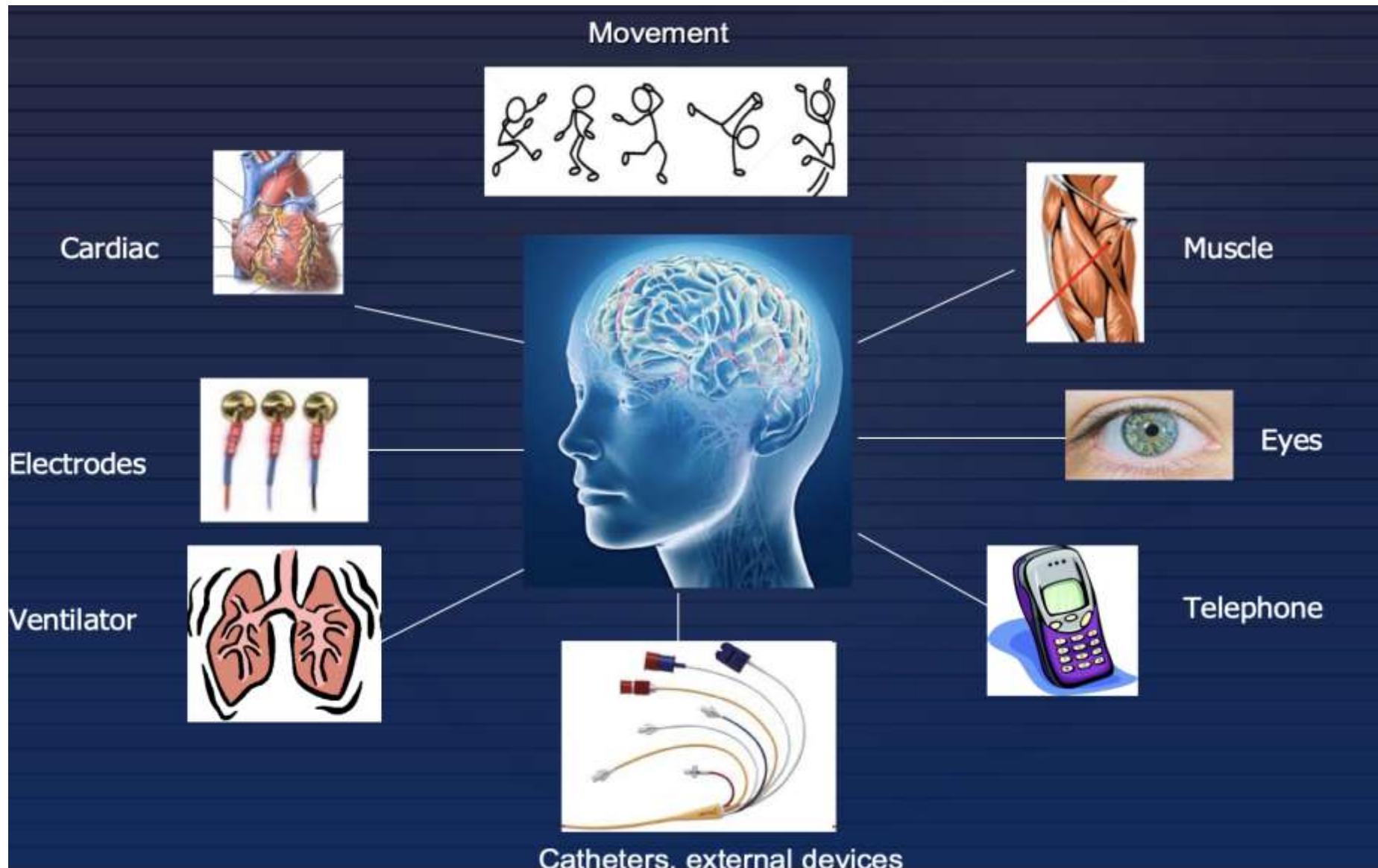
Principles to discriminate artifacts from EEG signals

- Physiological activity has a logical topographic field of distribution with an expected fall of the voltage potentials
- Artifact have an illogical distribution that defies the principles of localization

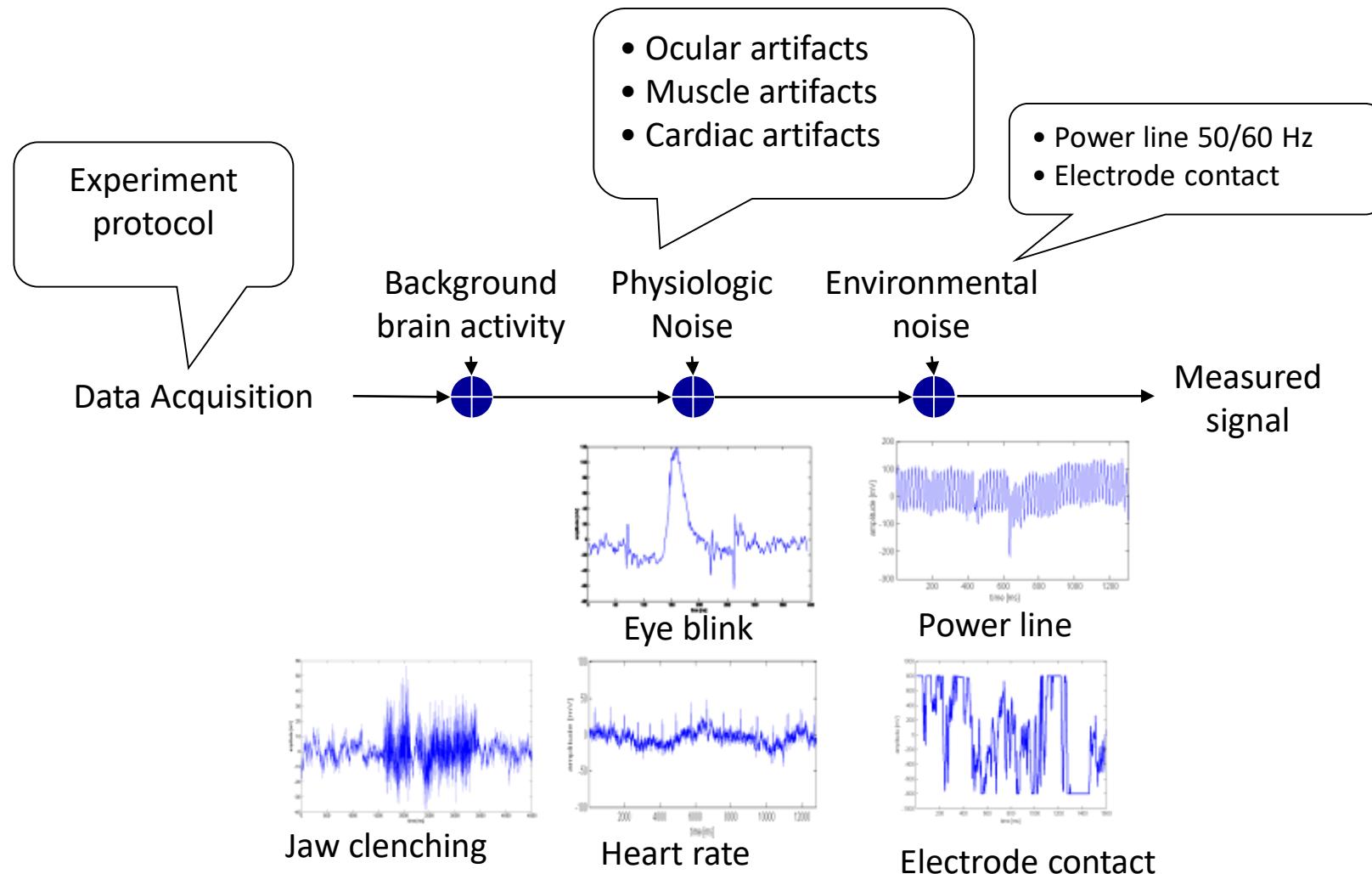
KEY TO AN ARTIFACT FREE RECORDING

- Good, clean preparation
- Good hook-up, neatly bundled electrodes
- Place jack-box close to patients head
- Keep the subject cool, not cold
- Unplug all electrical items close to patient, i.e. bed, radio, fan, etc.

EEG Artifacts

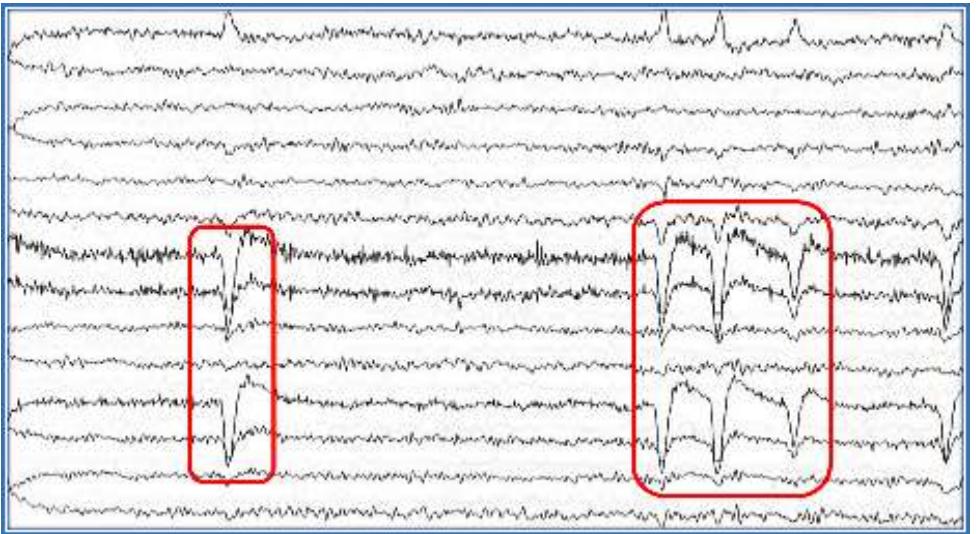


Captured EEG Signal

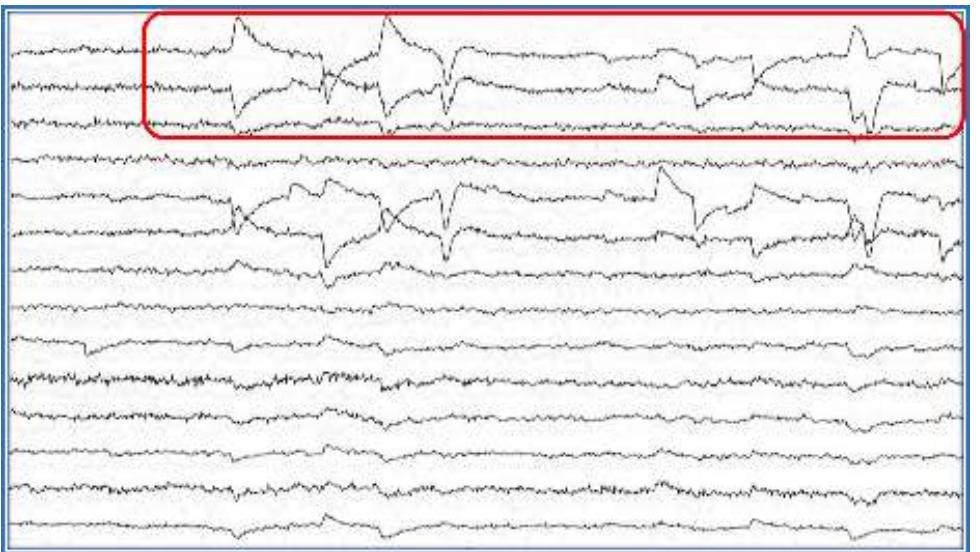


Ocular Artifacts

- Blinks
- Eye flutter
- Lateral gaze
- Slow/roving eye movement
- Rapid eye movement
- Electroretinogram (ERG)



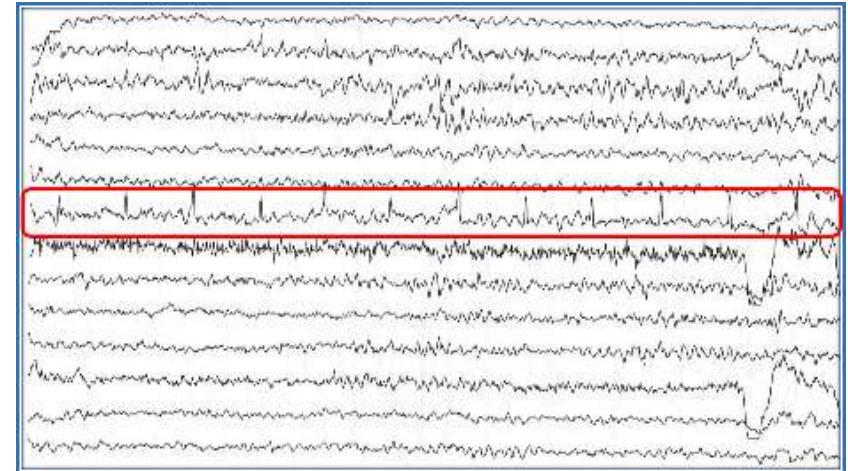
Blink



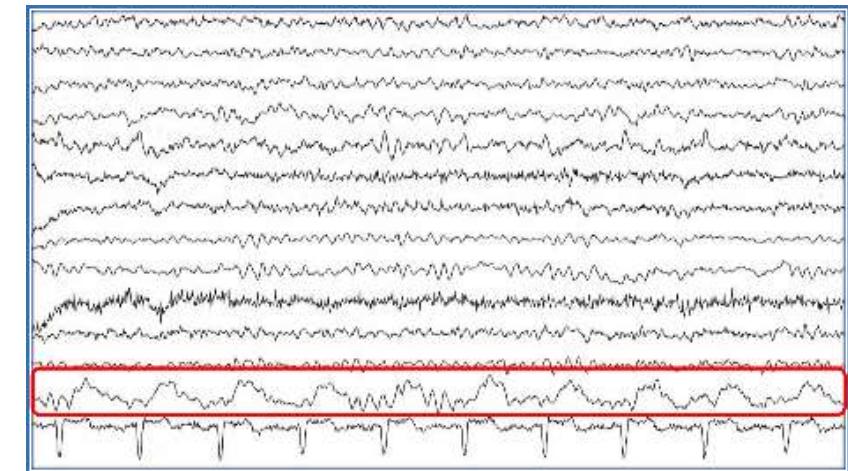
Lateral Eye Movement

Cardiac Artifacts

- Mechanical and Electrical
- ECG, Pacemaker - Electrical
- Pulse, Ballistocardiographic – Mechanical
- ✓ Mostly these are high in amplitude and prominent in babies, obese and short neck persons.
- ✓ Referential montages picks up cardiac artifacts.



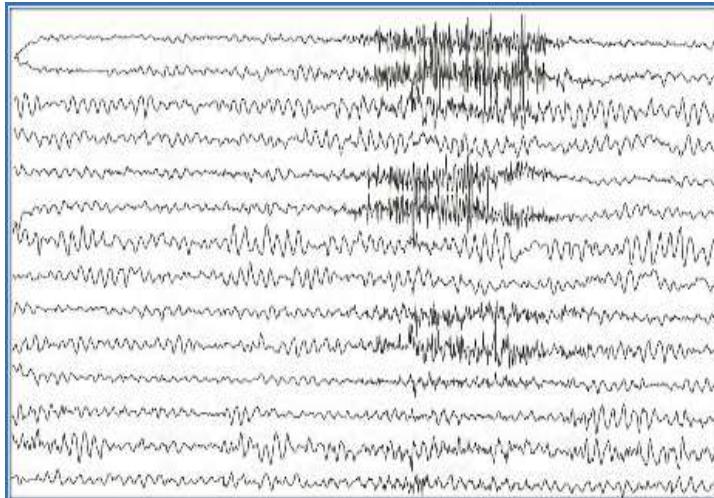
Cardiac (Electrical)



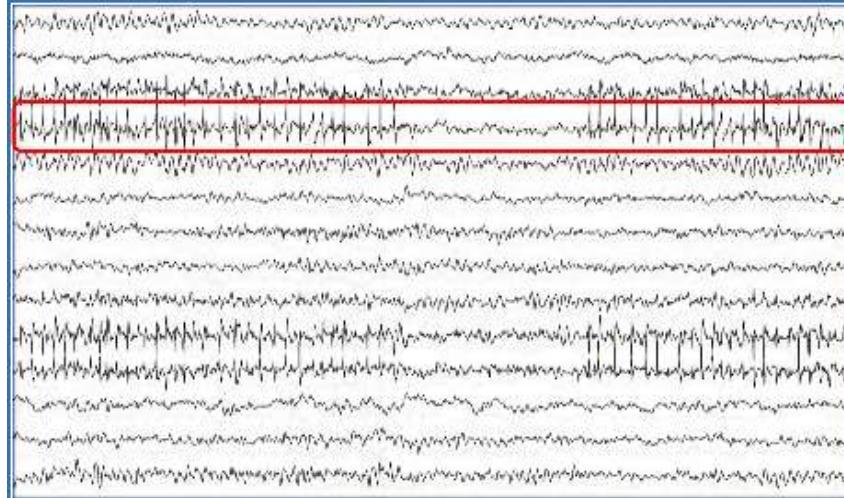
Cardiac (Mechanical)

Muscle Artifacts

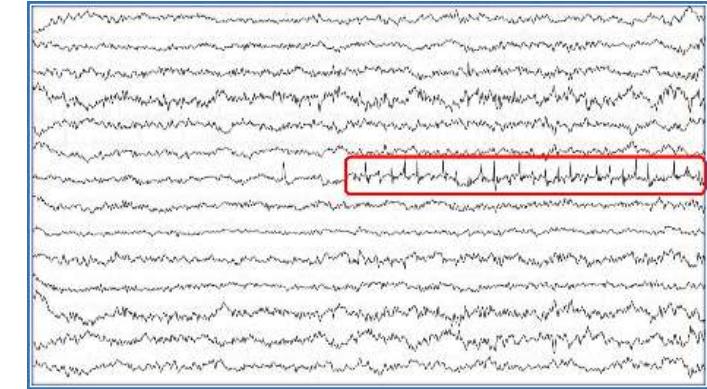
- ✓ Glossokenetic (related to tongue movements, Chew and swallow)
- ✓ Photomyogenic/ Photo-myoclonic (When flash of light falls over the face, the activity occurs due to myoclonus of the facial muscles).
- ✓ Surface EMG (Electromyography) – used to measure electrical activity during muscle contractions and relaxation cycles.



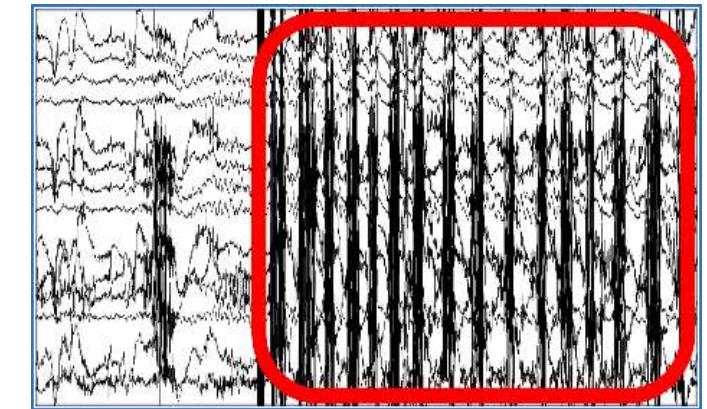
Electromyography (Scalp)



Electromyography (Facial)



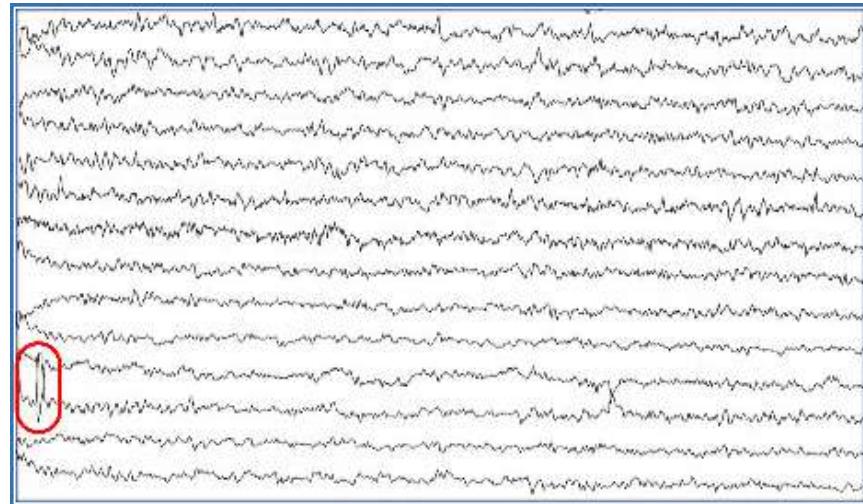
Photomyogenic



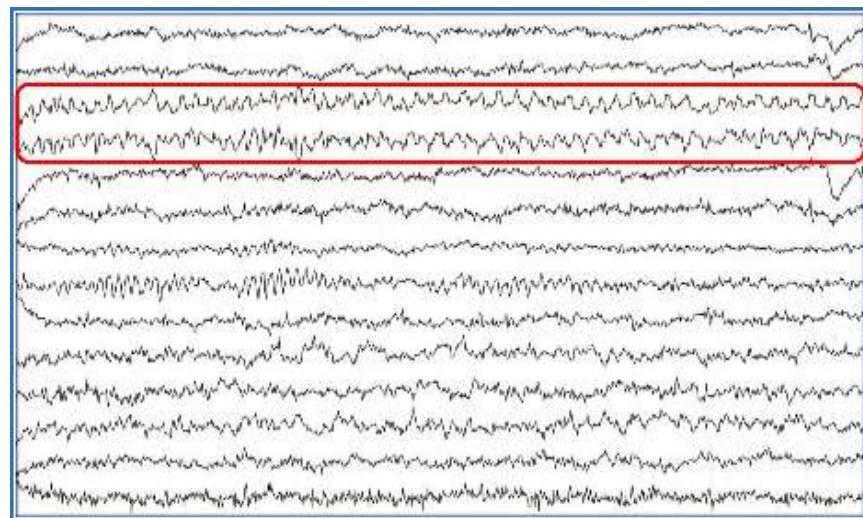
Chewing

Electrode and Equipment Artifacts

- ✓ Electrode pop, electrode contact, electrode movement
- ✓ Perspiration – the process of sweating
- ✓ salt bridge – differs from perspiration by low amplitude.
- ✓ Movement artifacts - Movement of head, body and limbs produce irregular high voltage potentials
- ✓ 50/60 Hz ambient electrical noise.
- ✓ Ventilators, circulatory pumps.
- ✓ Telephone, mobile.



Electrode Pop



Electrode Movement

Electrode and Equipment Artifacts

- Seen due to smearing of the electrode paste between electrodes or presence of perspiration across the scalp
- Forms an unwanted electrical connection between the electrodes forming a channel

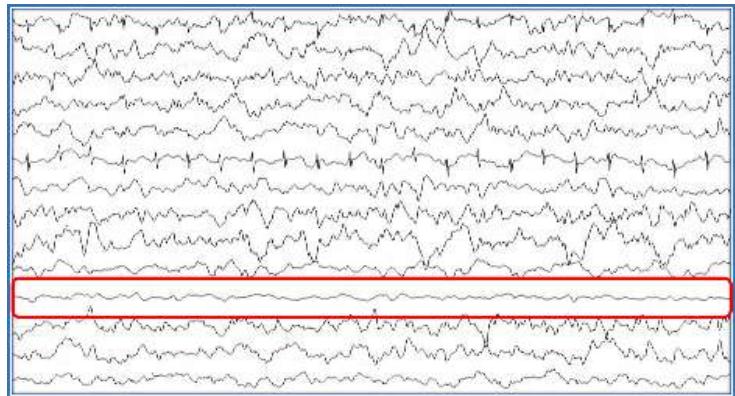
✓ Perspiration artifact

- manifests as low amplitude
- undulating (smooth) waves
- duration is typically greater than 2 sec

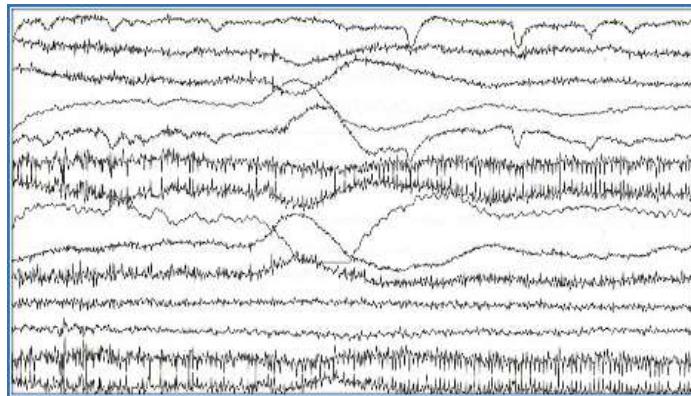
✓ Slat bridge artifact

- lower in amplitude
- not wavering with low frequency oscillation - typically include only one channel
- It may appear flat and close to isoelectric

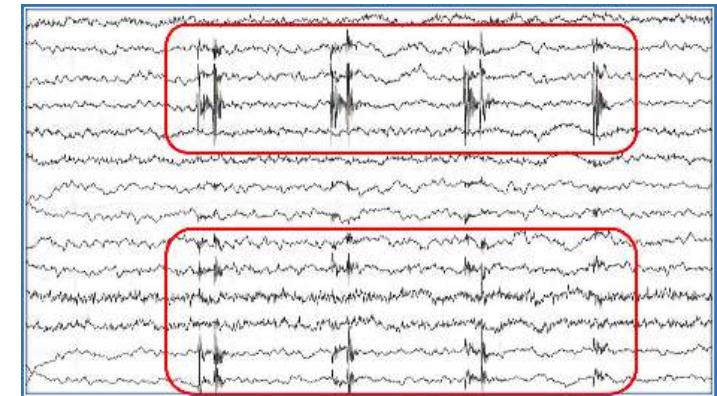
Electrode and Equipment Artifacts



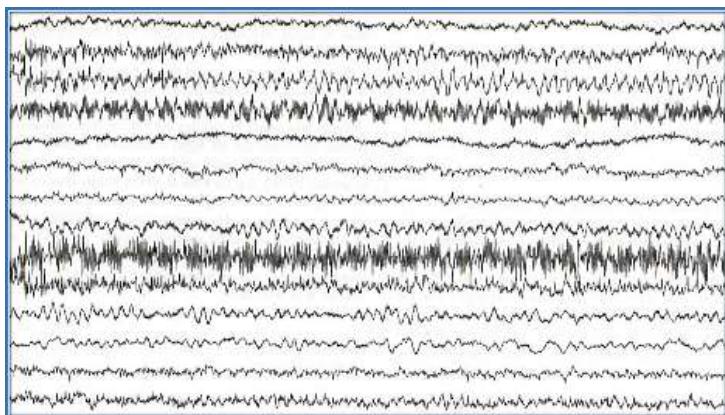
Salt Bridge



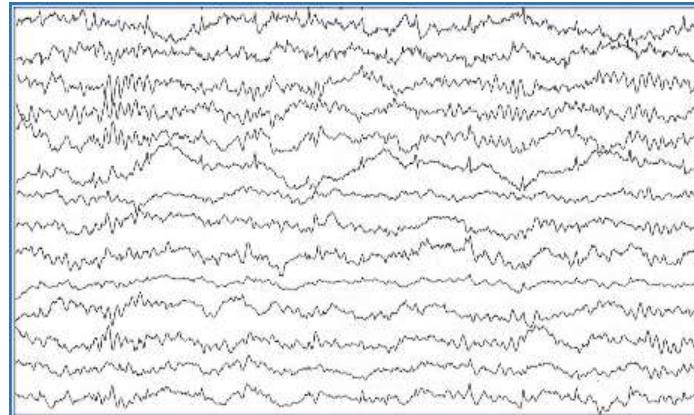
Electrode Lead Movement



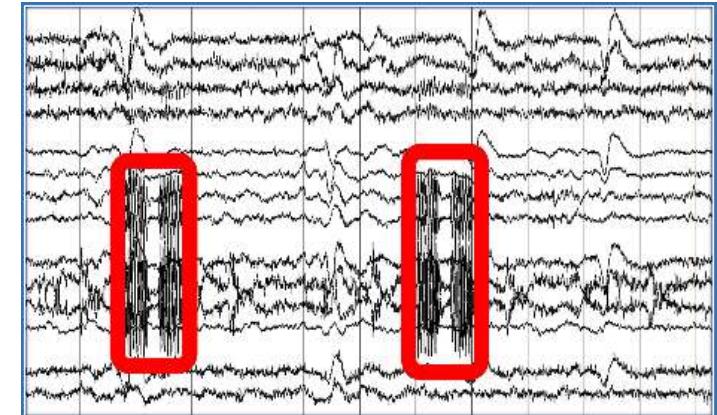
Electrical Motor



60 Hz



Perspiration



Phone

Thank You!

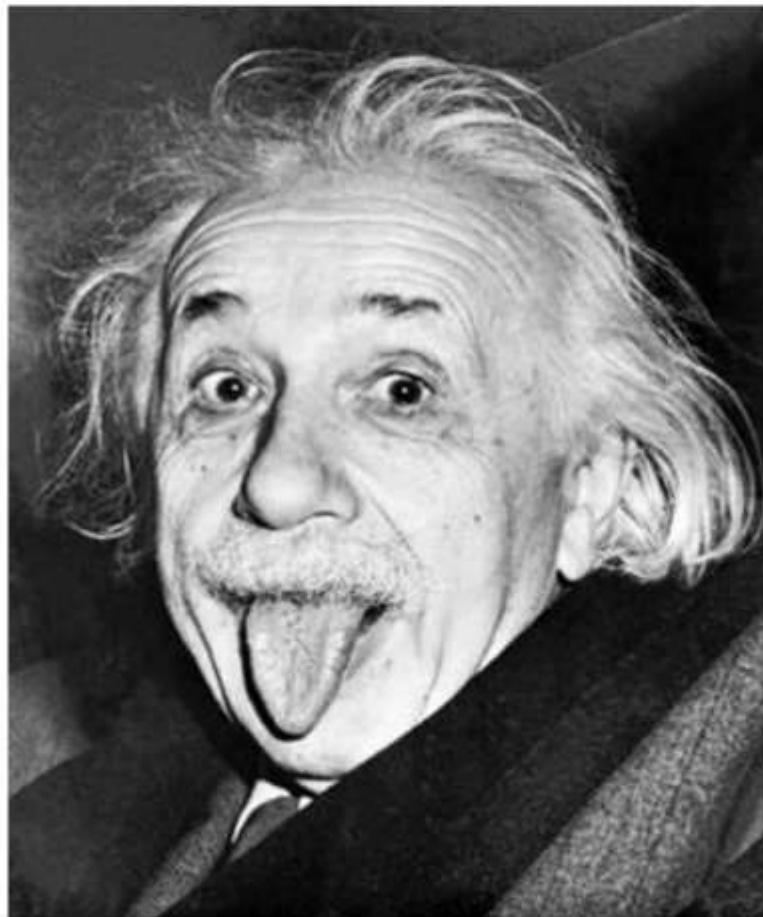


EEG Signal Pre-processing - Epoching

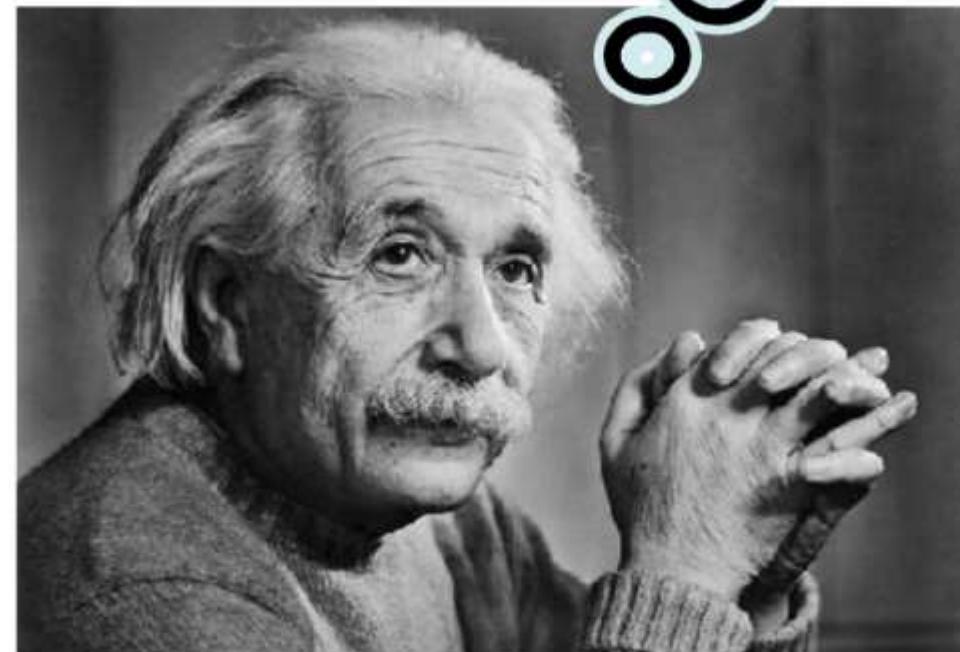
Course Instructor

Dr. Annushree Bablani

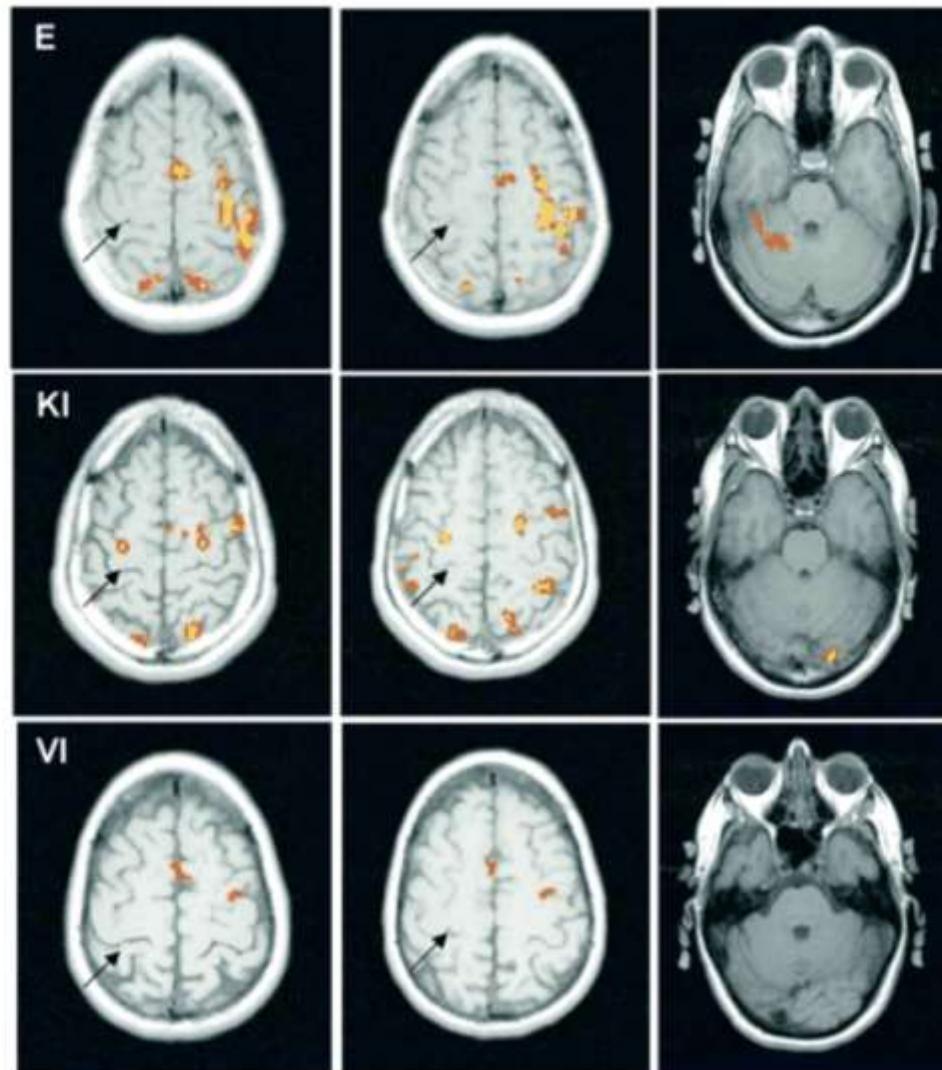
Motor Imagery



VS

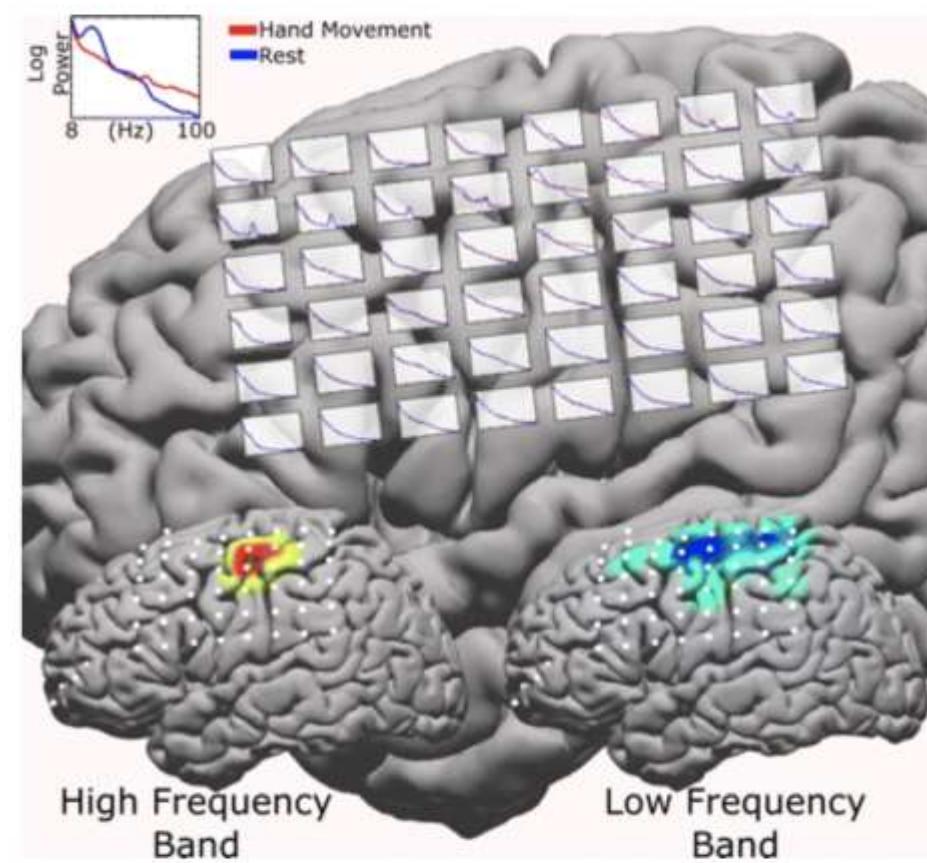
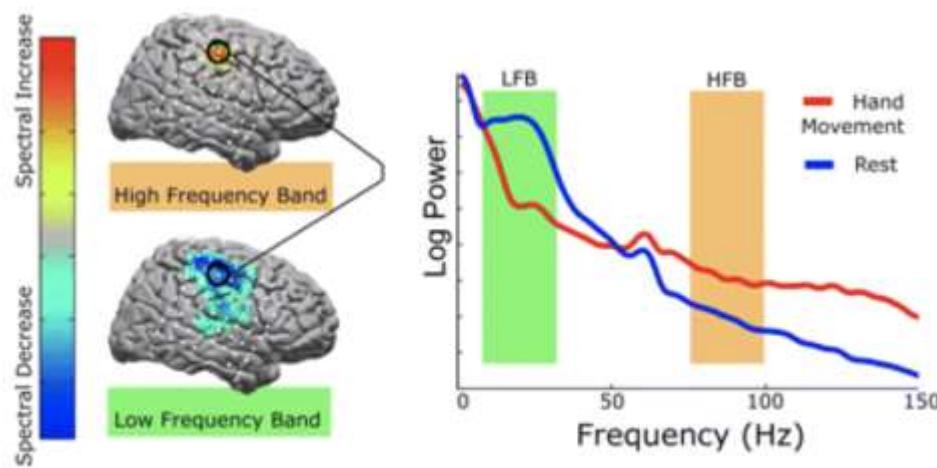
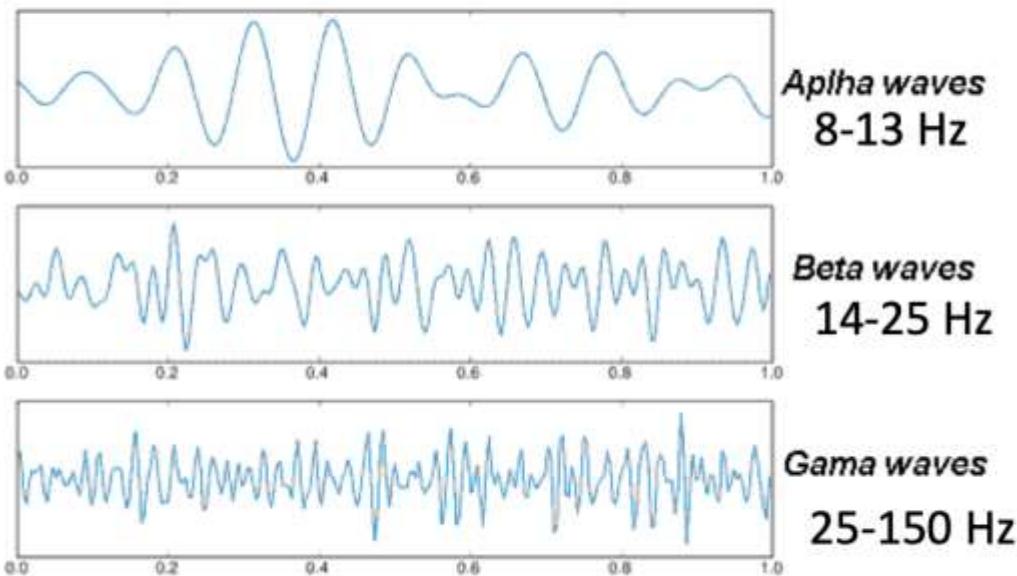


Brain activity during motor action vs imagery



Solodkin et al, 2004

Motor rhythms in cortex



Miller et al, 2007

Goal of the work

To check whether real movements can be compared with imagery movements.

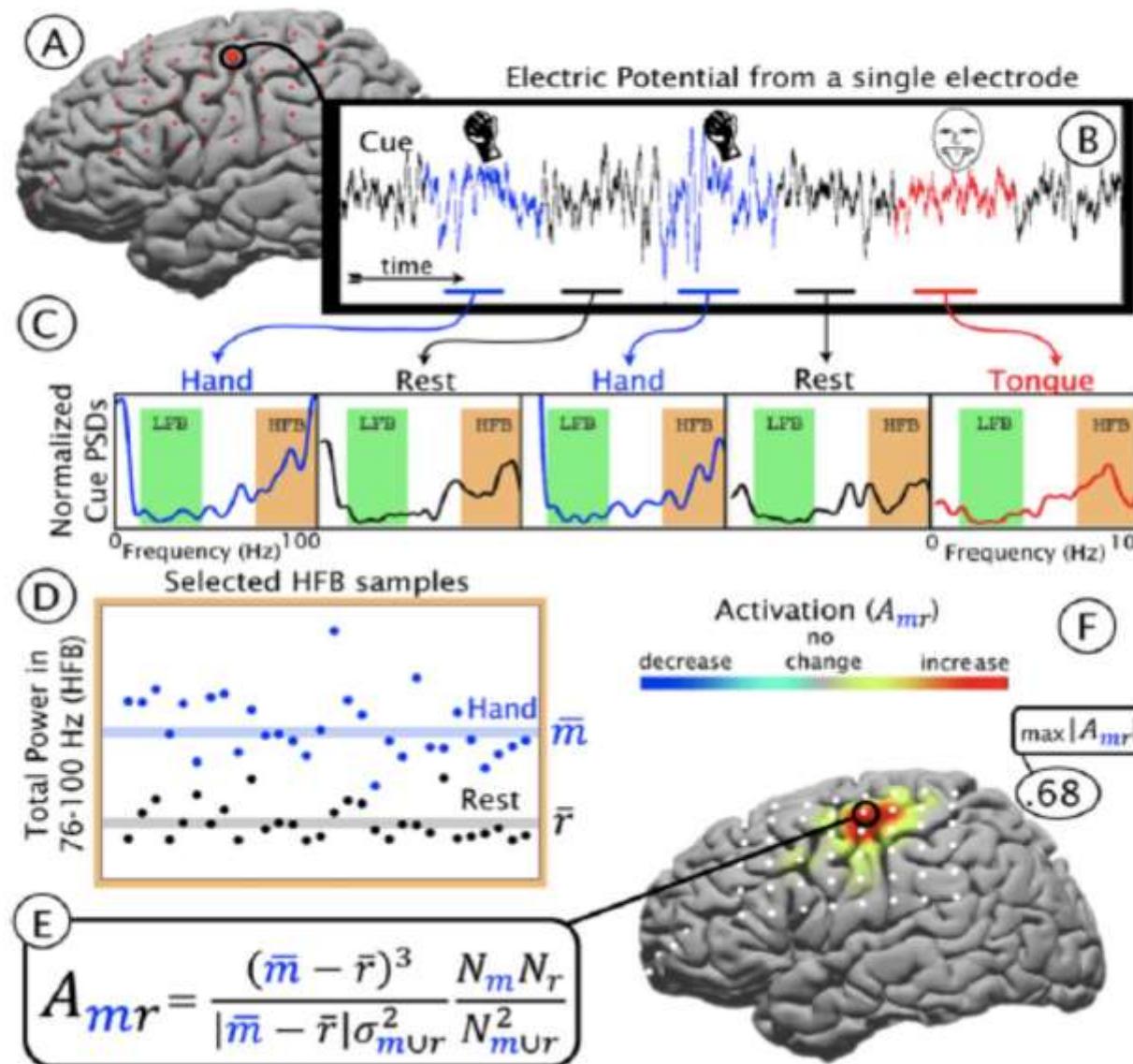
Methods Used:

- 8 patients implanted with 4x8 or 8x8 grid

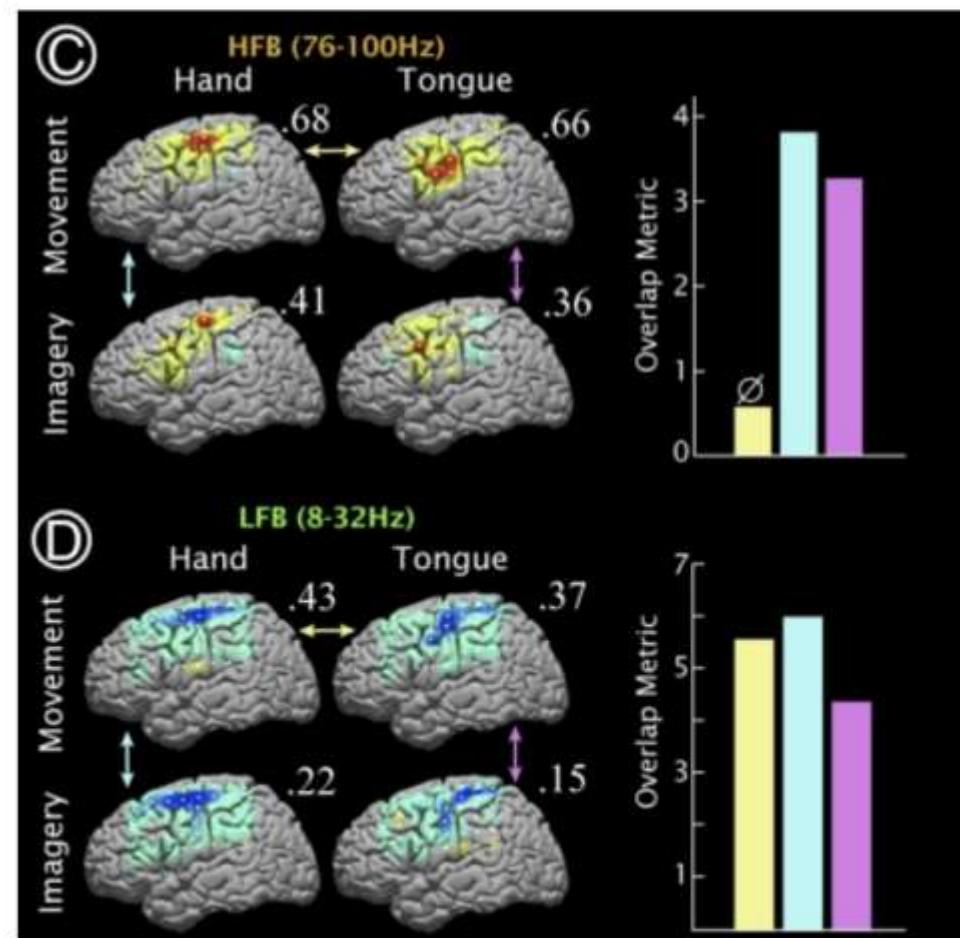
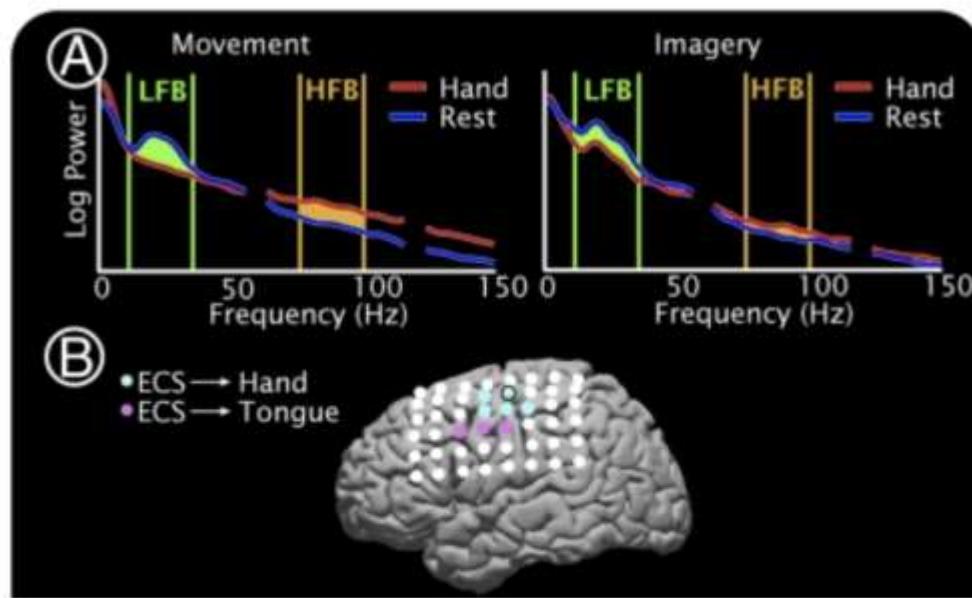
1) Active movements –clench – release of hand, stick out tongue, shoulder shrug, say the word “move”

2) Image movements – same as before

Quantification of brain activity



Cortical activity during real and imagined movements



Data Epoching

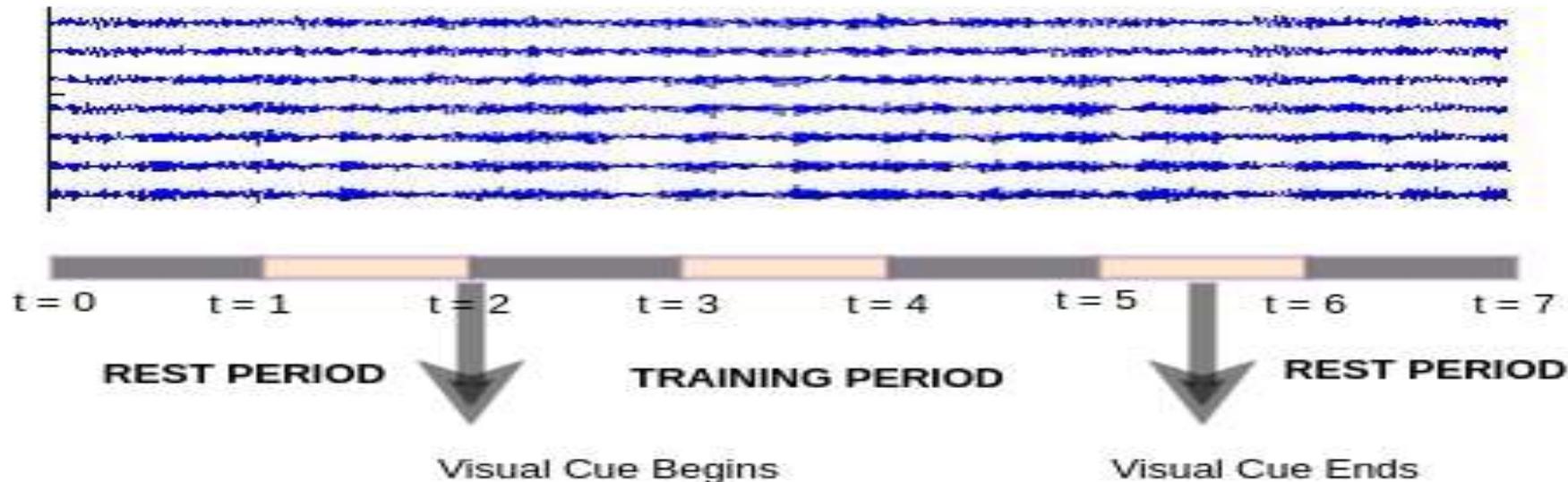
Selecting data epochs

There is no real good reason to select subsets of data epochs. When comparing conditions – performed by creating contrast at the STUDY level (the group analysis interface which may also be used for single-subject analysis) – one may ignore specific data epochs.

- Non-overlapping segments
- Overlapping segments
 - Fully overlapping segments
 - Partially overlapping segments

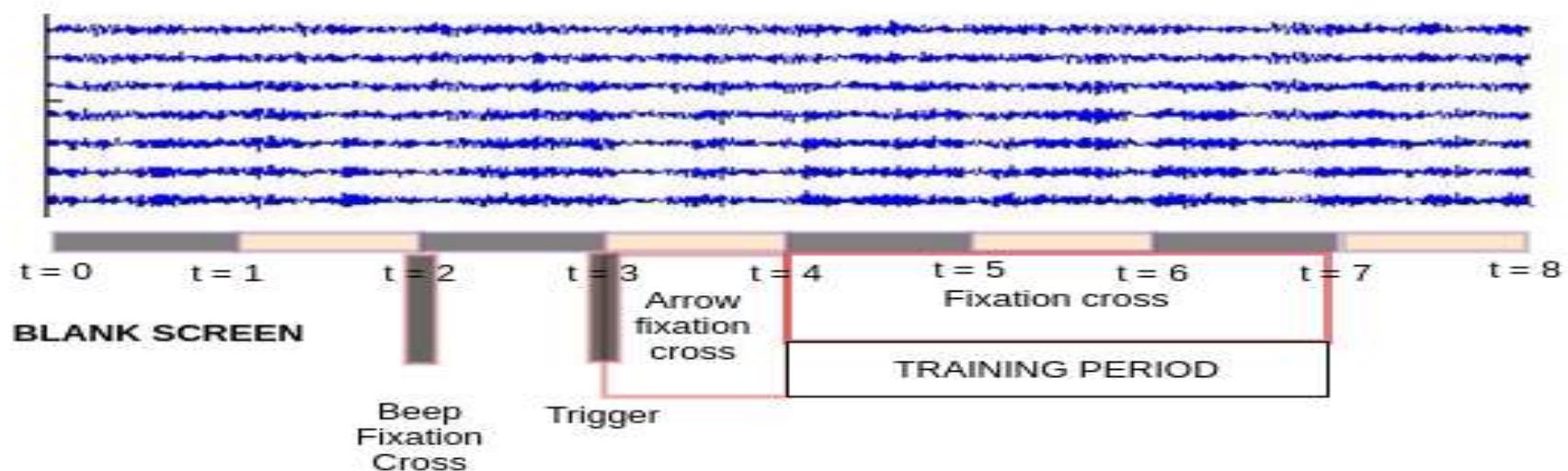
DATASET 1 SINGLE TRIAL

Class : Right hand / Foot

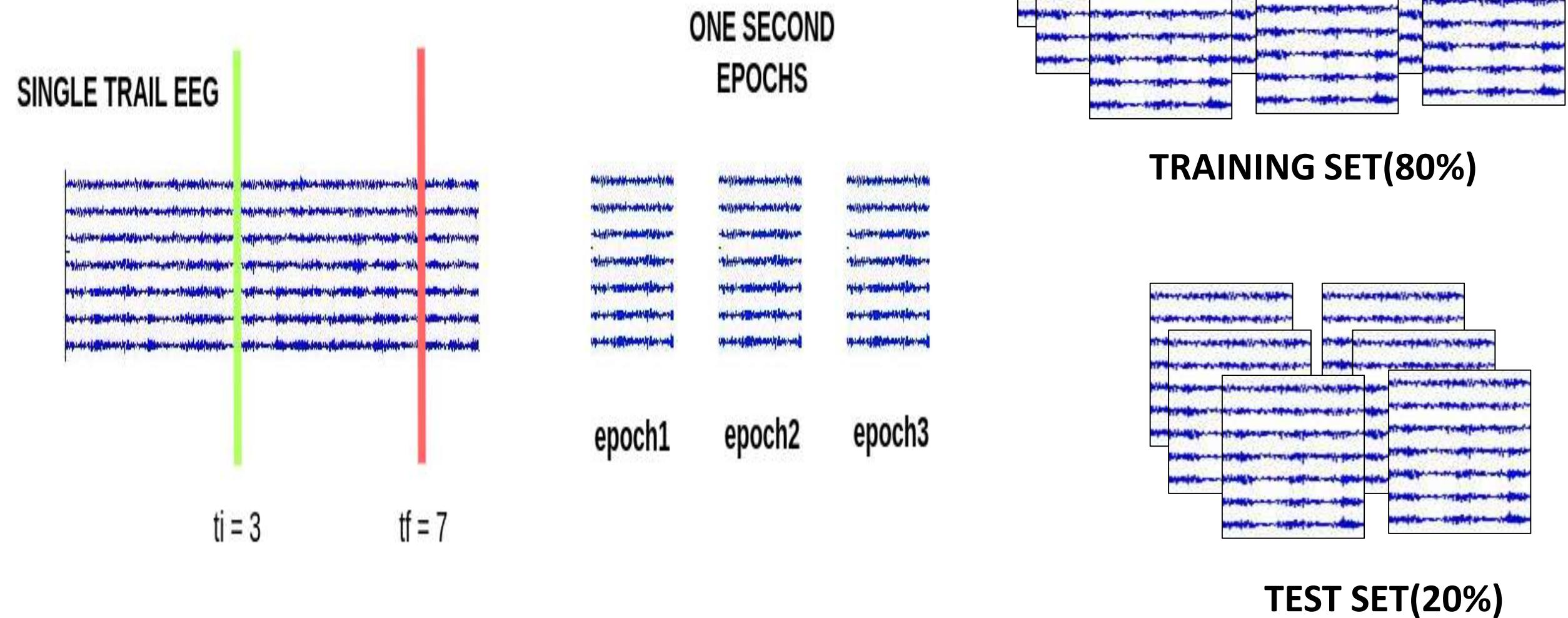


DATASET 2 SINGLE TRIAL

Class : Right hand / Left hand / Foot / Tongue



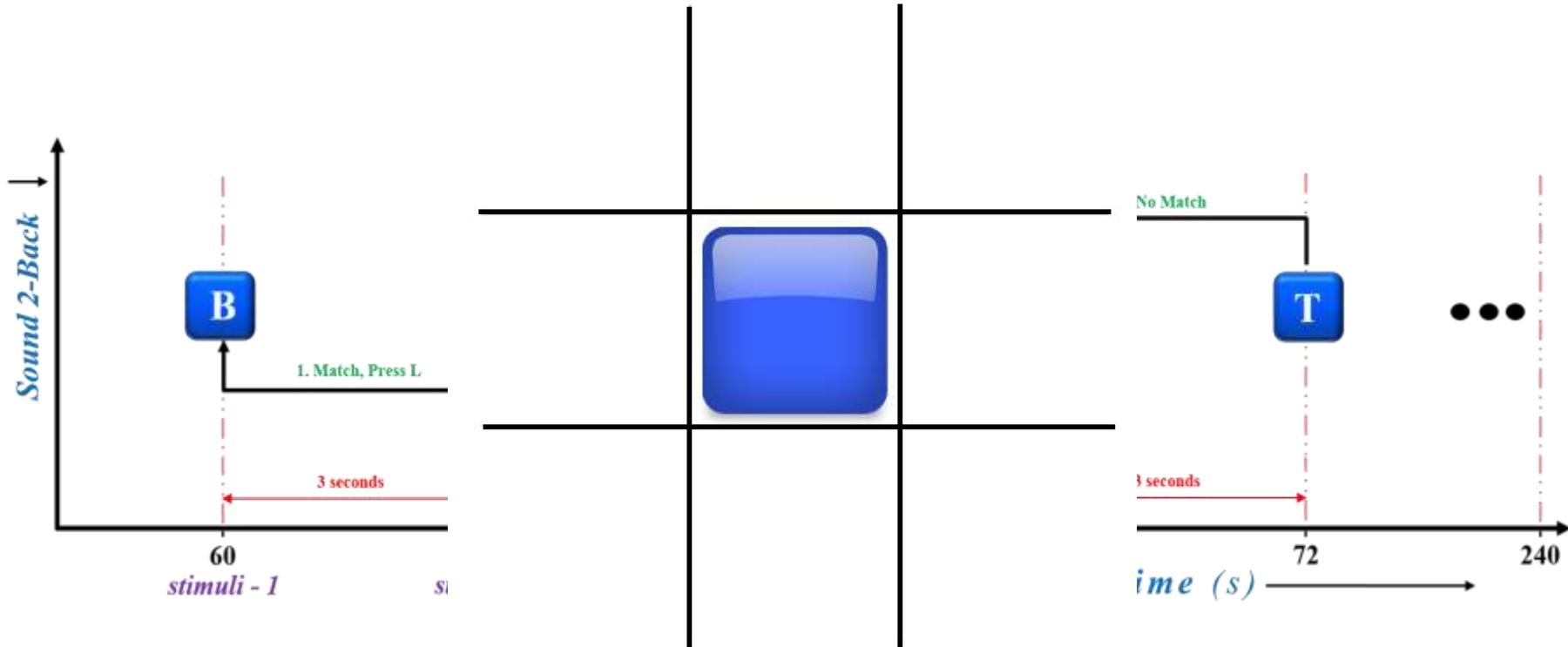
Data Epoching



Cognitive Workload

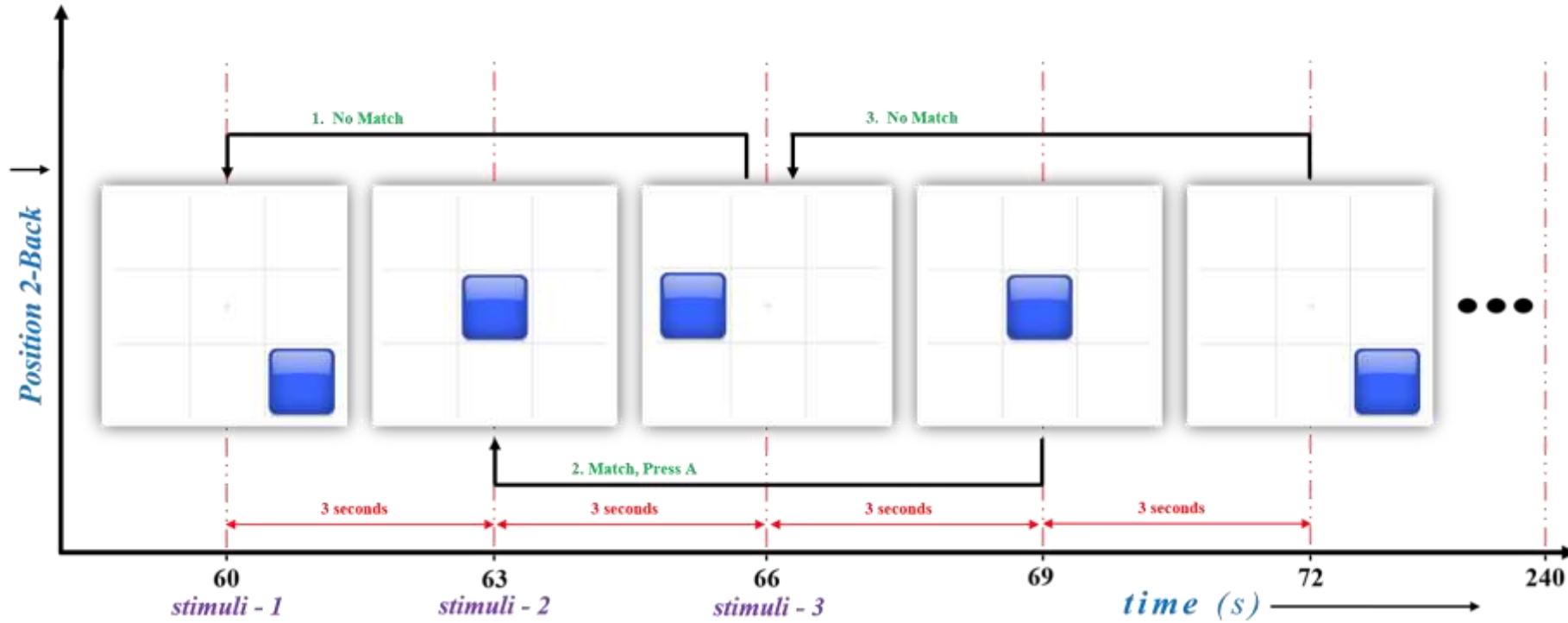
- Cognitive Workload (**CWL**) is a demand placed upon humans for mental resources while performing a task.
- Mental resources include working memory, ability to process, etc.
- **Working memory (WM)** is a cognitive system with a limited capacity to hold a small amount of information and process it.

n-Back Task



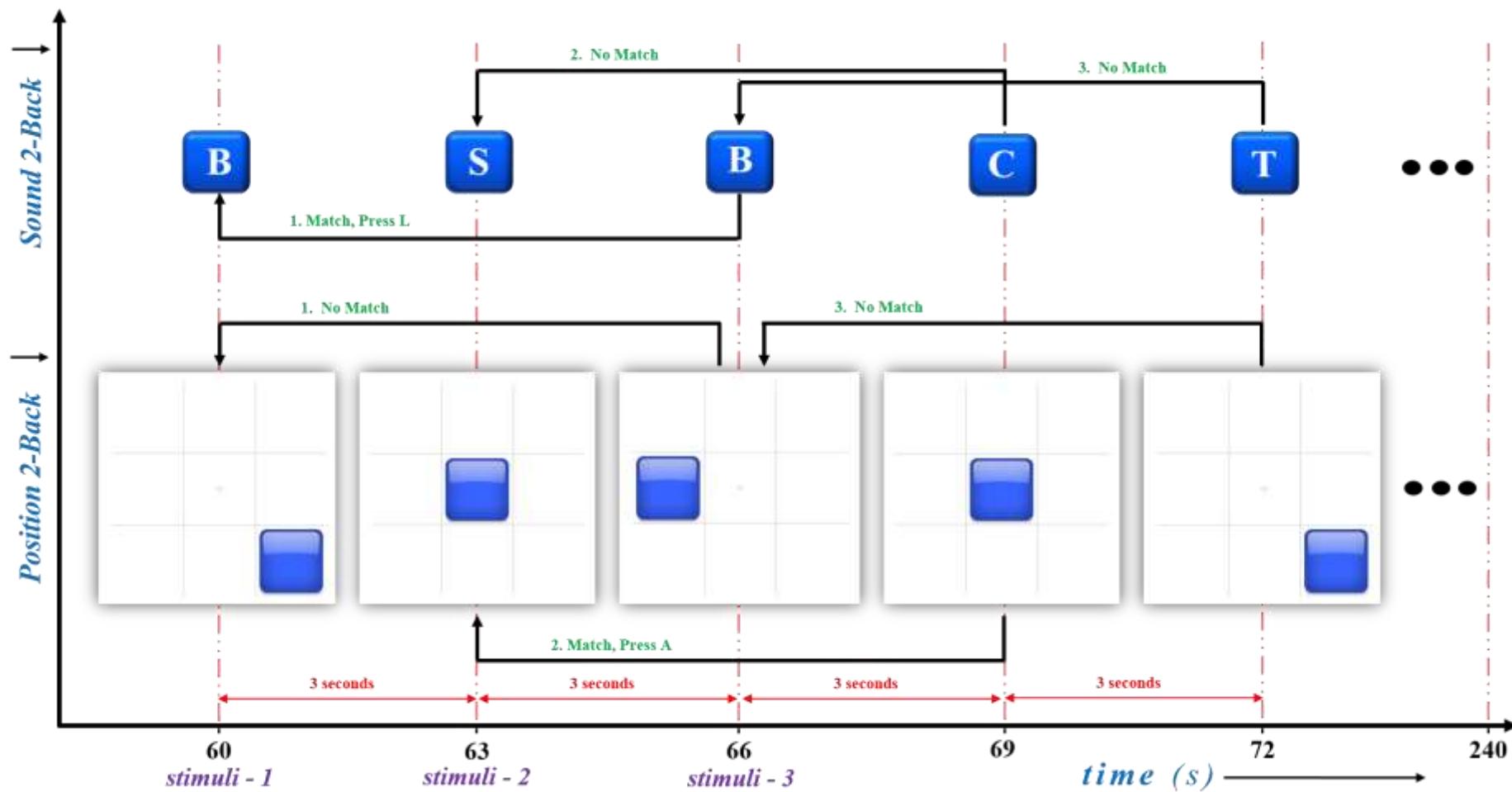
Sound 2-Back

n-Back Task



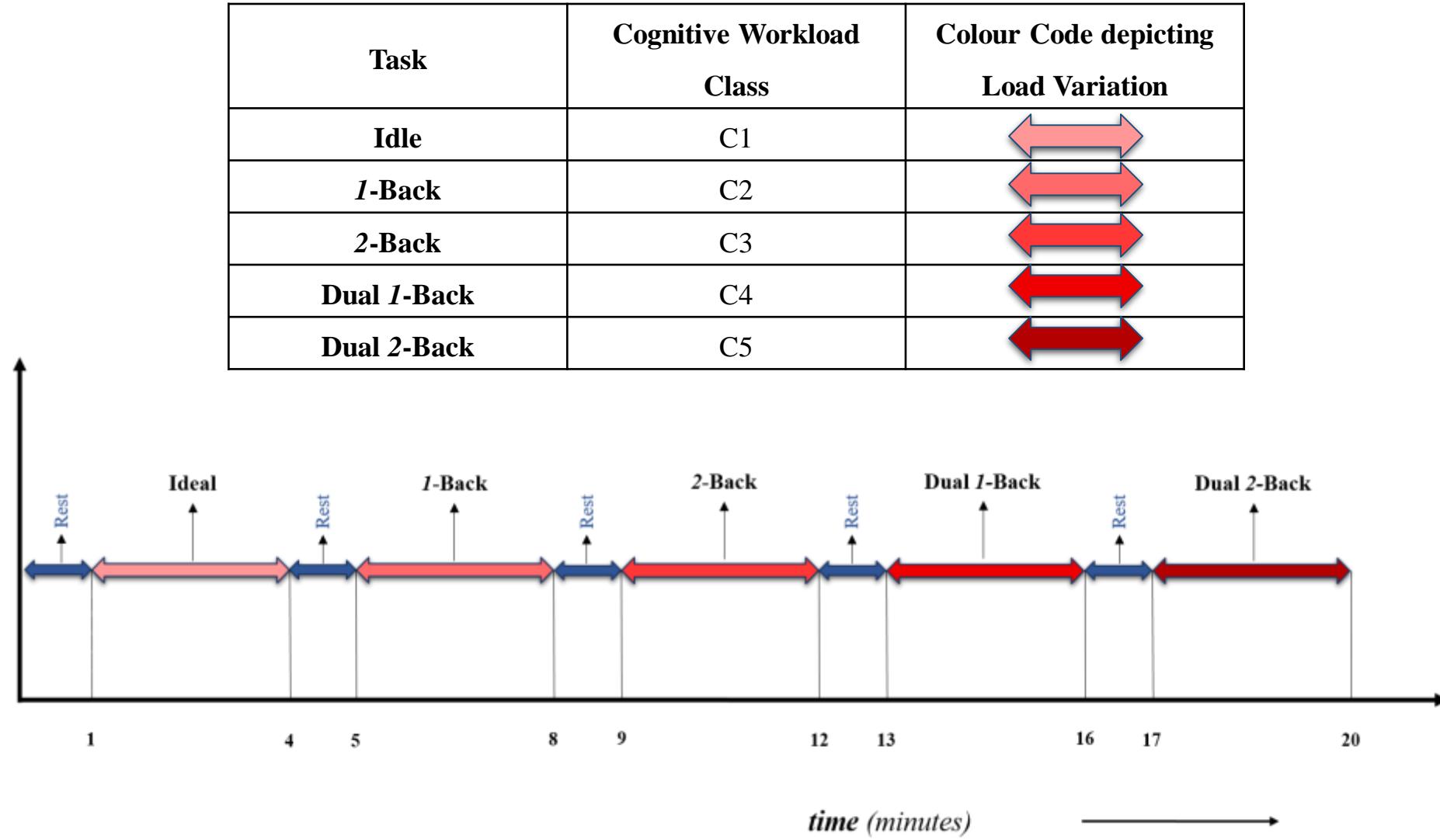
Position 2-Back

n-Back Task

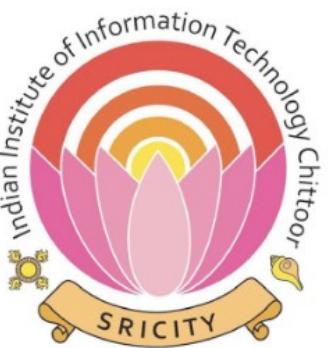


Sound & Position 2-Back

Data Epoching



Thank you!!



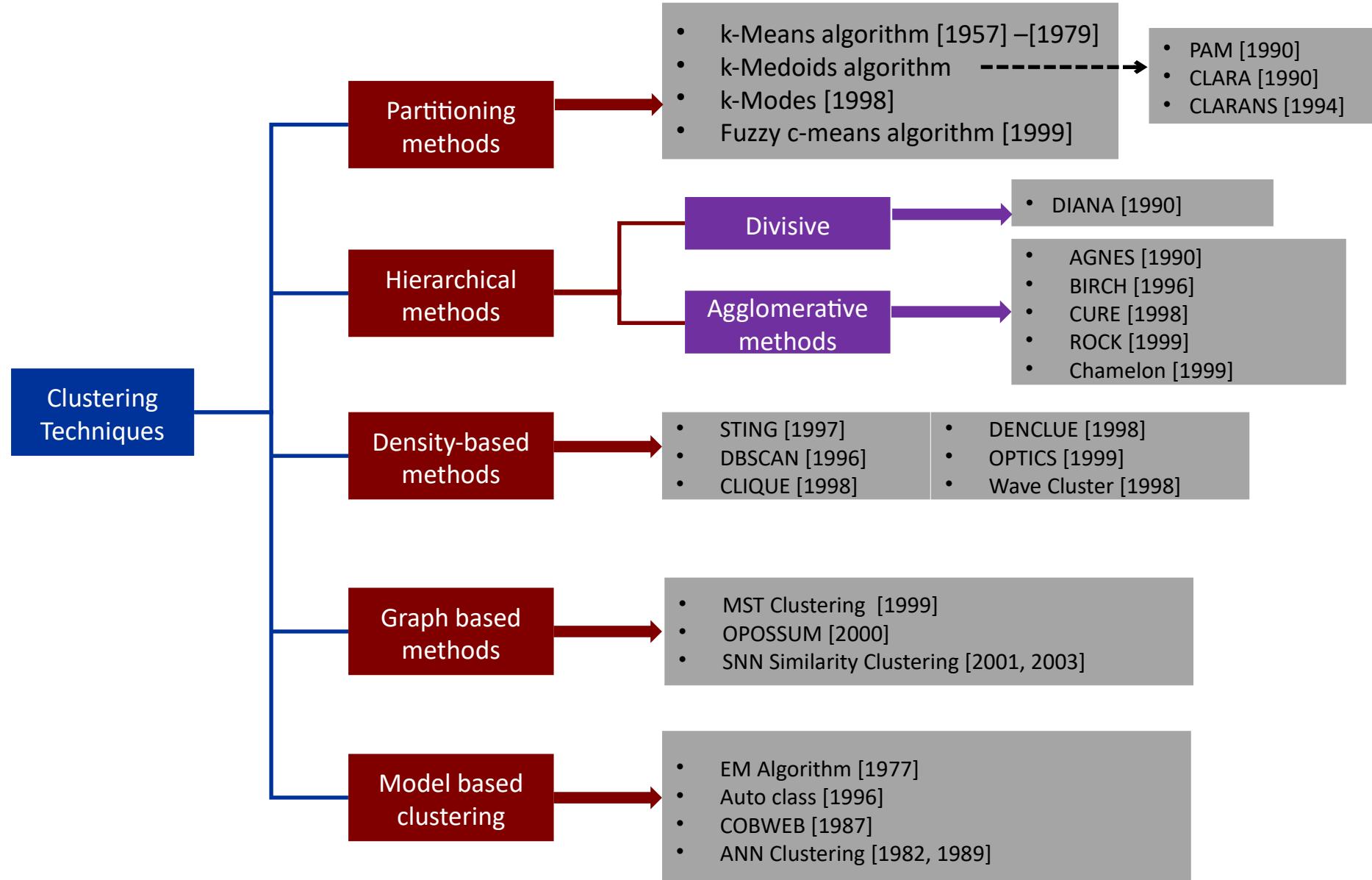
Brain Computer Interaction

Module III
Clustering techniques

**Indian Institute of Information Technology
IIIT Sri City**

Clustering techniques

- Clustering has been studied extensively for more than 40 years and across many disciplines due to its broad applications.
- As a result, many clustering techniques have been reported in the literature.
- Let us categorize the clustering methods. In fact, it is difficult to provide a crisp categorization because many techniques overlap to each other in terms of clustering paradigms or features.
- A broad taxonomy of existing clustering methods is shown in the next slide.
- It is not possible to cover all the techniques in this lecture series. We emphasize on major techniques belong to partitioning and hierarchical algorithms.



k-Means Algorithm

- k-Means clustering algorithm proposed by J. Hartigan and M. A. Wong [1979].
- Given a set of n distinct objects, the k-Means clustering algorithm partitions the objects into k number of clusters such that intraclass similarity is high but the interclass similarity is low.
- In this algorithm, user has to specify k , the number of clusters and consider the objects are defined with numeric attributes and thus using any one of the distance metric to demarcate the clusters.

k-Means Algorithm

The algorithm can be stated as follows.

- First it selects k number of objects at random from the set of n objects. These k objects are treated as the **centroids or center of gravities** of k clusters.
- For each of the **remaining objects**, it is assigned to one of the **closest centroid**. Thus, it forms a **collection of objects assigned to each centroid** and is called a **cluster**.
- Next, the centroid of each cluster is then updated (by calculating the mean values of attributes of each object).
- The assignment and update procedure is until it reaches some stopping criteria (such as, number of iteration, centroids remain unchanged or no assignment, etc.)

k-Means Algorithm

Algorithm 24.1: k-Means clustering

Input: D is a dataset containing n objects, k is the number of cluster

Output: A set of k clusters

Steps:

1. Randomly choose k objects from D as the initial cluster centroids.
2. **For** each of the objects in D **do**
 - Compute distance between the current objects and k cluster centroids
 - Assign the current object to that cluster to which it is closest.
3. Compute the “cluster centers” of each cluster. These become the new cluster centroids.
4. Repeat step 2-3 until the convergence criterion is satisfied
5. Stop

k-Means Algorithm

Note:

- 1) Objects are defined in terms of set of attributes. where each is continuous data type.
- 2) Distance computation: Any distance such as or cosine similarity.
- 3) Minimum distance is the measure of closeness between an object and centroid.
- 4) Mean Calculation: It is the mean value of each attribute values of all objects.
- 5) Convergence criteria: Any one of the following are termination condition of the algorithm.
 - Number of maximum iteration permissible.
 - No change of centroid values in any cluster.
 - Zero (or no significant) movement(s) of object from one cluster to another.
 - Cluster quality reaches to a certain level of acceptance.

Illustration of k-Means clustering algorithms

Table 24.1: 16 objects with two attributes and .

A ₁	A ₂
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1

Fig 24.1: Plotting data of Table 24.1

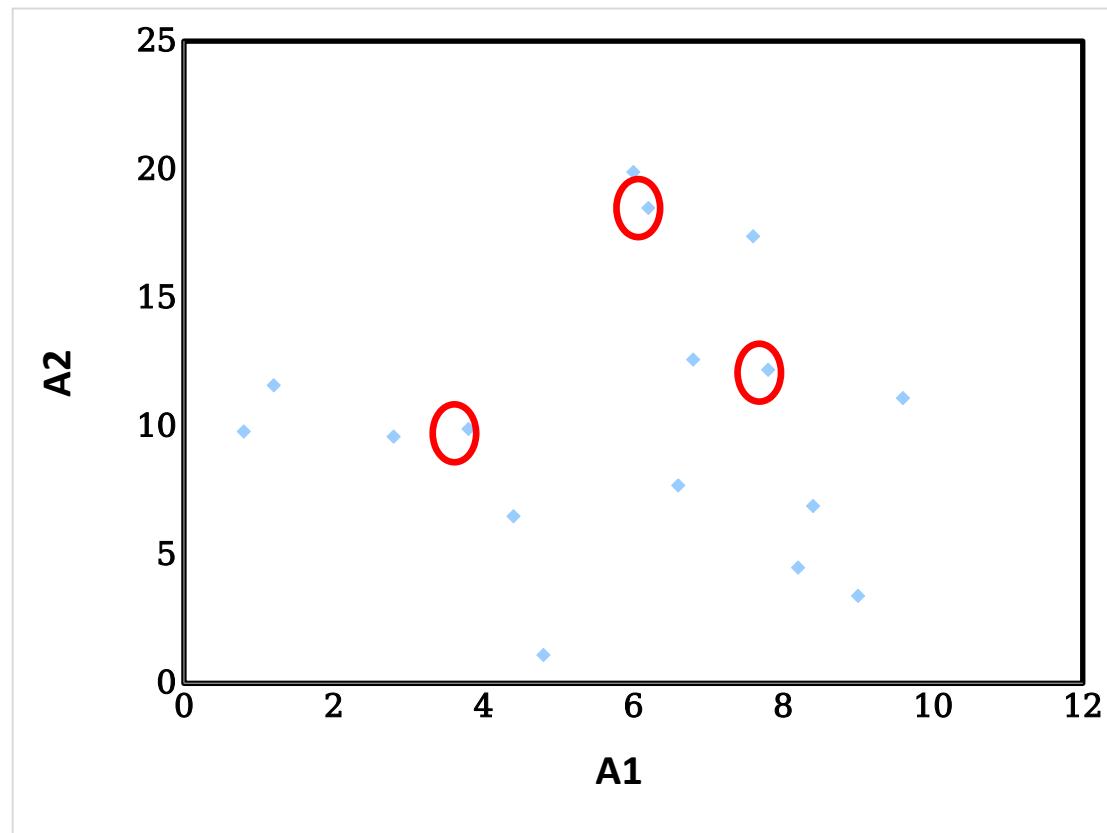


Illustration of k-Means clustering algorithms

- Suppose, $k=3$. Three objects are chosen at random shown as circled (see Fig 24.1). These three centroids are shown below.

Initial Centroids chosen randomly

Centroid	Objects	
c_1	3.8	9.9
c_2	7.8	12.2
c_3	6.2	18.5

- Let us consider the Euclidean distance measure (L_2 Norm) as the distance measurement in our illustration.
- Let d_1 , d_2 and d_3 denote the distance from an object to c_1 , c_2 and c_3 respectively. The distance calculations are shown in Table 24.2.
- Assignment of each object to the respective centroid is shown in the right-most column and the clustering so obtained is shown in Fig 24.2.

Illustration of k-Means clustering algorithms

Table 24.2: Distance calculation

A ₁	A ₂	d ₁	d ₂	d ₃	cluster
6.8	12. 6	4.0	1.1	5.9	2
0.8	9.8	3.0	7.4	10.2	1
1.2	11. 6	3.1	6.6	8.5	1
2.8	9.6	1.0	5.6	9.5	1
3.8	9.9	0.0	4.6	8.9	1
4.4	6.5	3.5	6.6	12.1	1
4.8	1.1	8.9	11.5	17.5	1
6.0	19. 9	10.2	7.9	1.4	3
6.2	18. 5	8.9	6.5	0.0	3
7.6	17. 4	8.4	5.2	1.8	3
7.8	12. 2	4.6	0.0	6.5	2
6.6	7.7	3.6	4.7	10.8	1
8.2	4.5	7.0	7.7	14.1	1
8.4	6.9	5.5	5.3	11.8	2
9.0	3.4	8.3	8.9	15.4	1
9.6	11. 1	5.9	2.1	8.1	2

Fig 24.2: Initial cluster with respect to Table 24.2

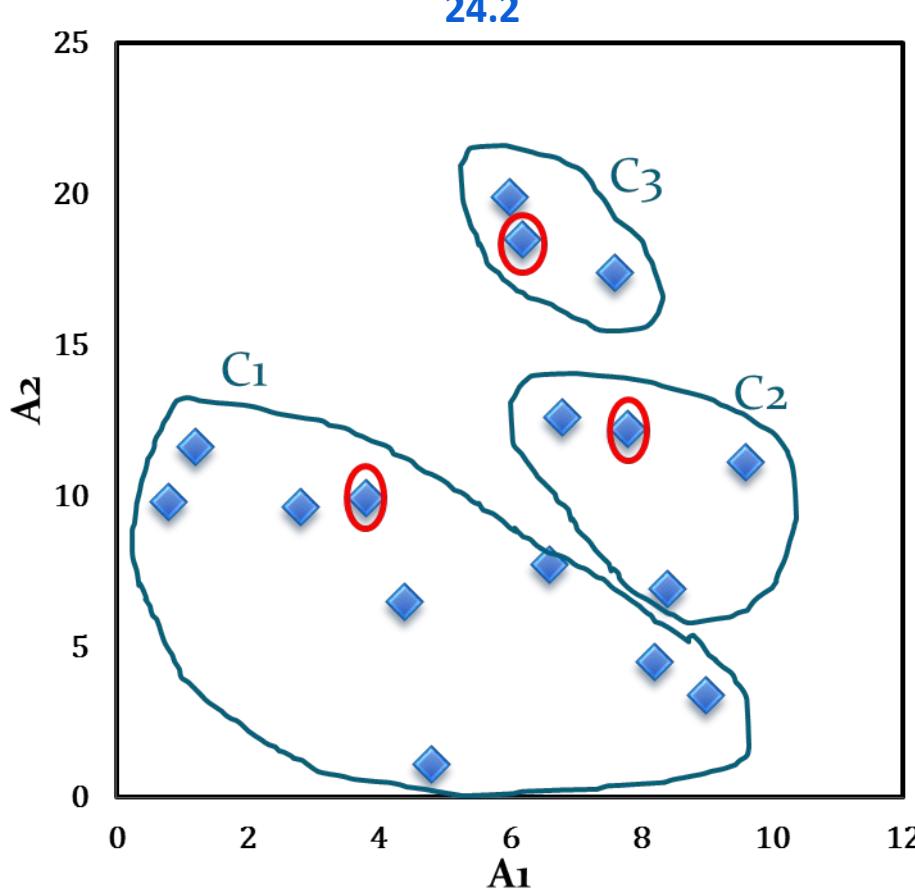


Illustration of k-Means clustering algorithms

The calculation new centroids of the three cluster using the mean of attribute values of A_1 and A_2 is shown in the Table below. The cluster with new centroids are shown in Fig 24.3.

Calculation of new centroids

New Centroi d	Objects	
c_1	4.6	7.1
c_2	8.2	10.7
c_3	6.6	18.6

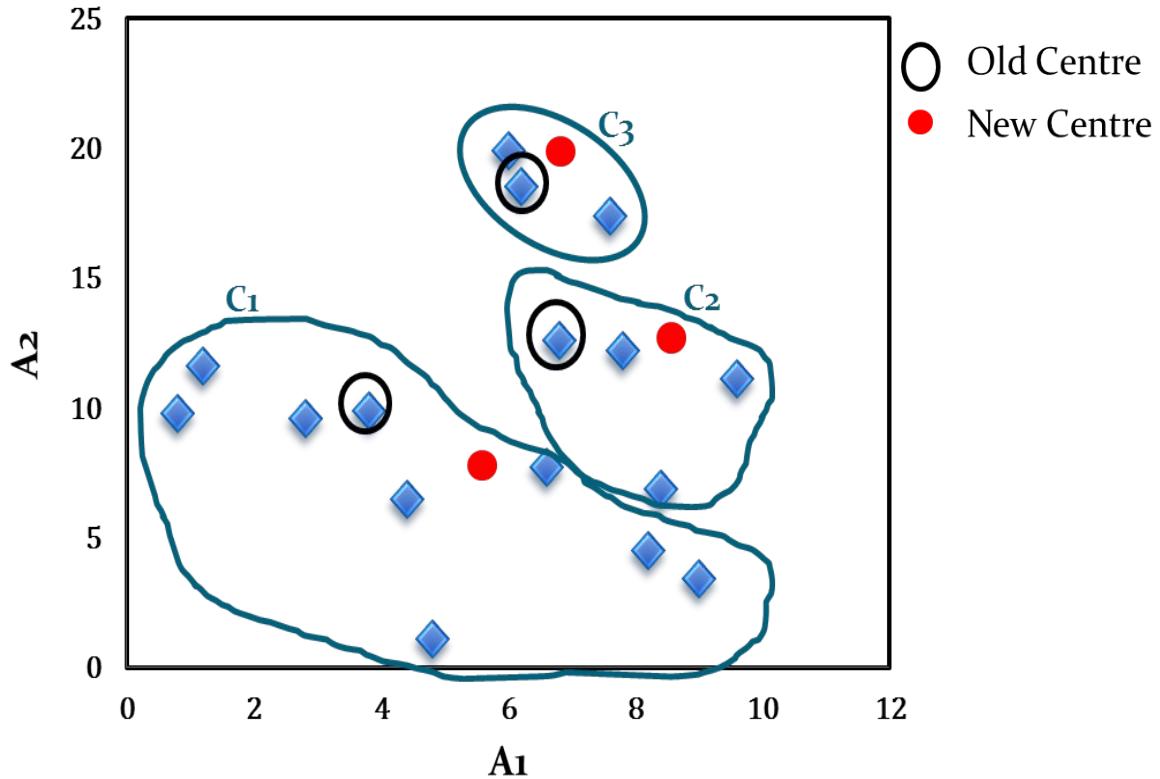


Fig 24.3: Initial cluster with new centroids

Illustration of k-Means clustering algorithms

We next reassign the 16 objects to three clusters by determining which centroid is closest to each one. This gives the revised set of clusters shown in Fig 24.4.

Note that point p moves from cluster C_2 to cluster C_1 .

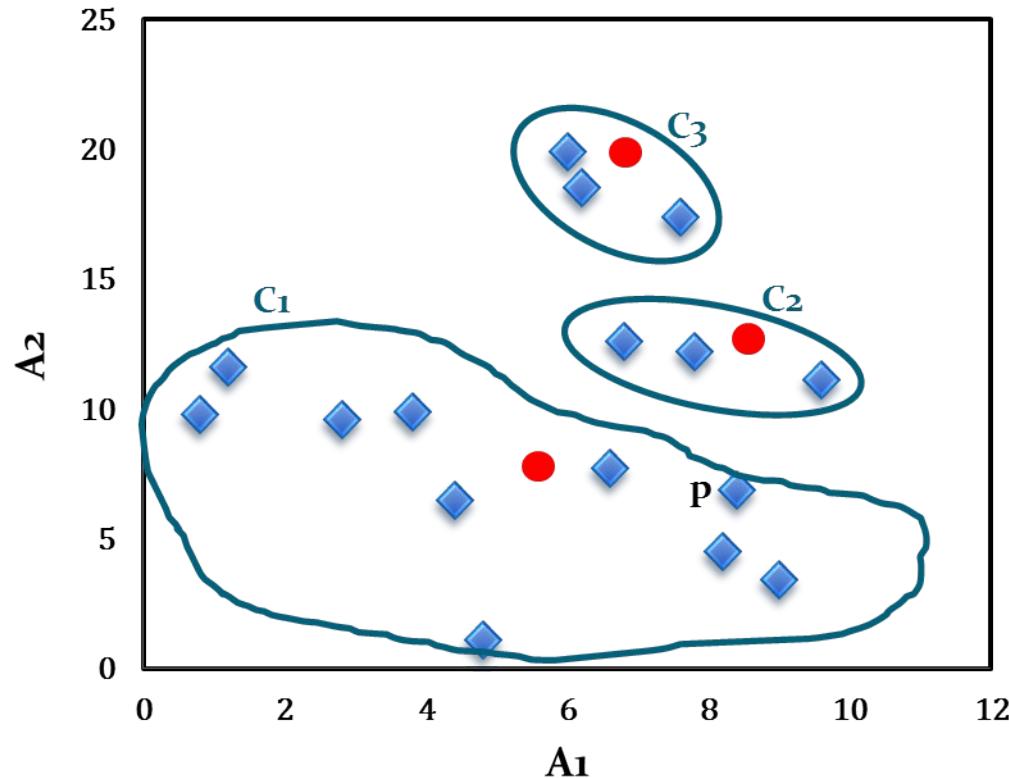
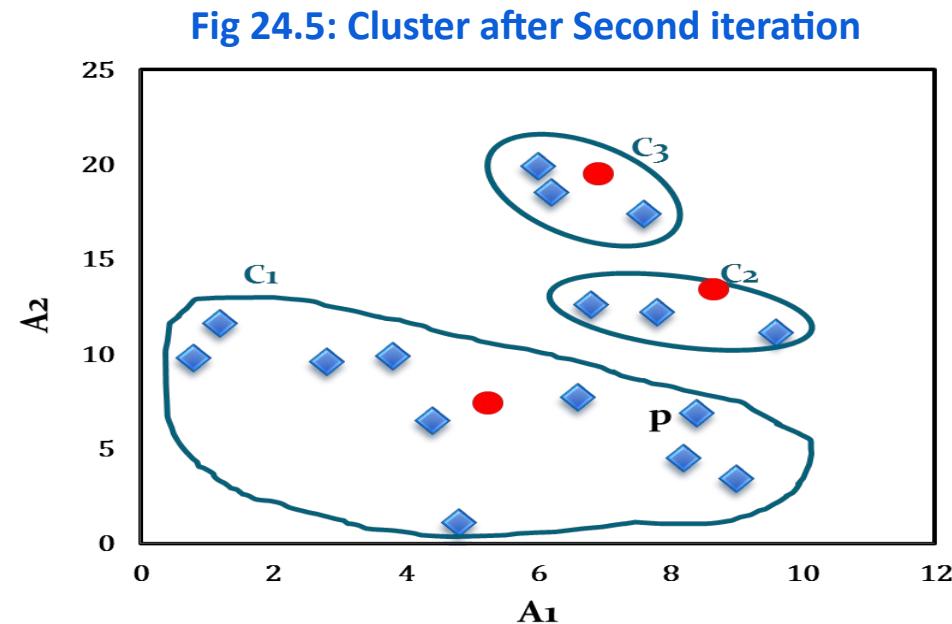


Fig 24.4: Cluster after first iteration

Illustration of k-Means clustering algorithms

- The newly obtained centroids after second iteration are given in the table below. Note that the centroid c_3 remains unchanged, where c_2 and c_1 changed a little.
- With respect to newly obtained cluster centres, 16 points are reassigned again. These are the same clusters as before. Hence, their centroids also remain unchanged.
- Considering this as the termination criteria, the k-means algorithm stops here. Hence, the final cluster in Fig 24.5 is same as Fig 24.4.

Cluster centres after second iteration		
Centroid	Revised Centroids	Centroids
c_1	5.0	7.1
c_2	8.1	12.0
c_3	6.6	18.6



Comments on k-Means algorithm

Advantages:

- k-Means is simple and can be used for a wide variety of object types.
- It is also efficient both from storage requirement and execution time point of views. By saving distance information from one iteration to the next, the actual number of distance calculations, that must be made can be reduced (specially, as it reaches towards the termination).

Limitations:

- The k-Means is not suitable for all types of data. For example, k-Means does not work on categorical data because mean cannot be defined.
- k-means finds a local optima and may actually minimize the global optimum.
- k-means has trouble clustering data that contains outliers.
- k-Means algorithm cannot handle non-globular clusters, clusters of different sizes and densities (see Fig 24.6 in the next slide).
- k-Means algorithm not really beyond the scalability issue (and not so practical for large databases).

Comments on k-Means algorithm

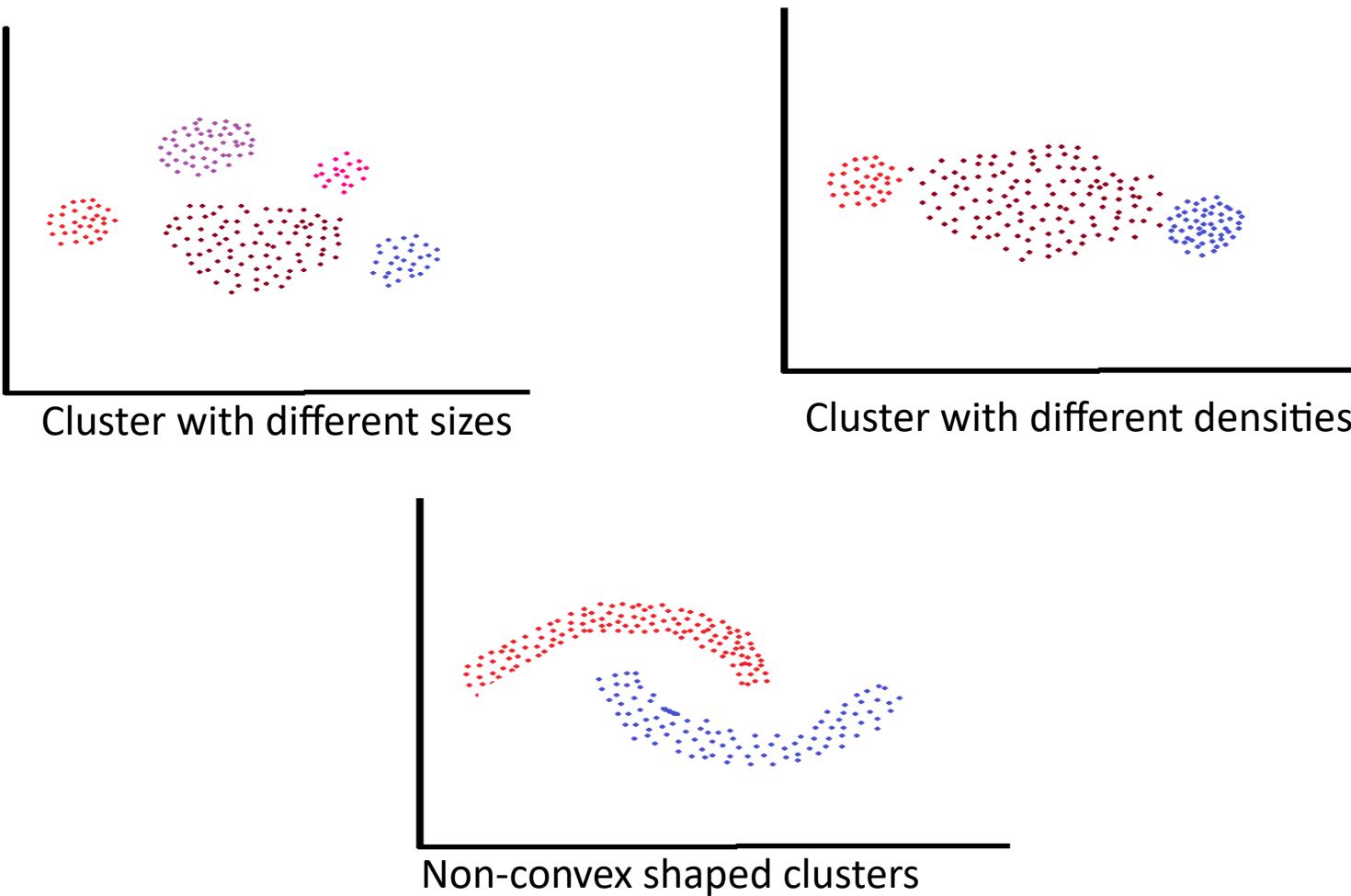


Fig 24.6: Some failure instance of k-Means algorithm

Different variants of k-means algorithm

There are a quite few variants of the k-Means algorithm. These can differ in the procedure of selecting the initial k means, the calculation of proximity and strategy for calculating cluster means. Another variants of k-means to cluster categorical data.

Few variant of k-Means algorithm includes

- Bisecting k-Means (addressing the issue of initial choice of cluster means).
 1. M. Steinbach, G. Karypis and V. Kumar “A comparison of document clustering techniques”, *Proceedings of KDD workshop on Text mining*, 2000.
- Mean of clusters (Proposing various strategies to define means and variants of means).
 - B. zhan “Generalised k-Harmonic means – Dynamic weighting of data in unsupervised learning”, *Technical report, HP Labs*, 2000.
 - A. D. Chaturvedi, P. E. Green, J. D. Carroll, “k-Modes clustering”, *Journal of classification*, Vol. 18, PP. 35-36, 2001.
 - D. Pelleg, A. Moore, “x-Means: Extending k-Means with efficient estimation of the number of clusters”, *17th International conference on Machine Learning*, 2000.

Different variants of k-means algorithm

- N. B. Karayiannis, M. M. Randolph, “Non-Euclidean c-Means clustering algorithm”, *Intelligent data analysis journal*, Vol 7(5), PP 405-425, 2003.
- V. J. Olivera, W. Pedrycy, “Advances in Fuzzy clustering and its applications”, Edited book. John Wiley [2007]. (Fuzzy c-Means algorithm).
- A. K. Jain and R. C. Bubes, “Algorithms for clustering Data”, Prentice Hall, 1988.
Online book at http://www.cse.msu.edu/~jain/clustering_Jain_Dubes.pdf
- A. K. Jain, M. N. Munty and P. J. Flynn, “Data clustering: A Review”, *ACM computing surveys*, 31(3), 264-323 [1999]. Also available online.

Any question?



Brain Computer Interaction

Performance Evaluation

Course Instructors

Dr. Annushree Bablani

Acknowledgements: Dr. Sreeja S R

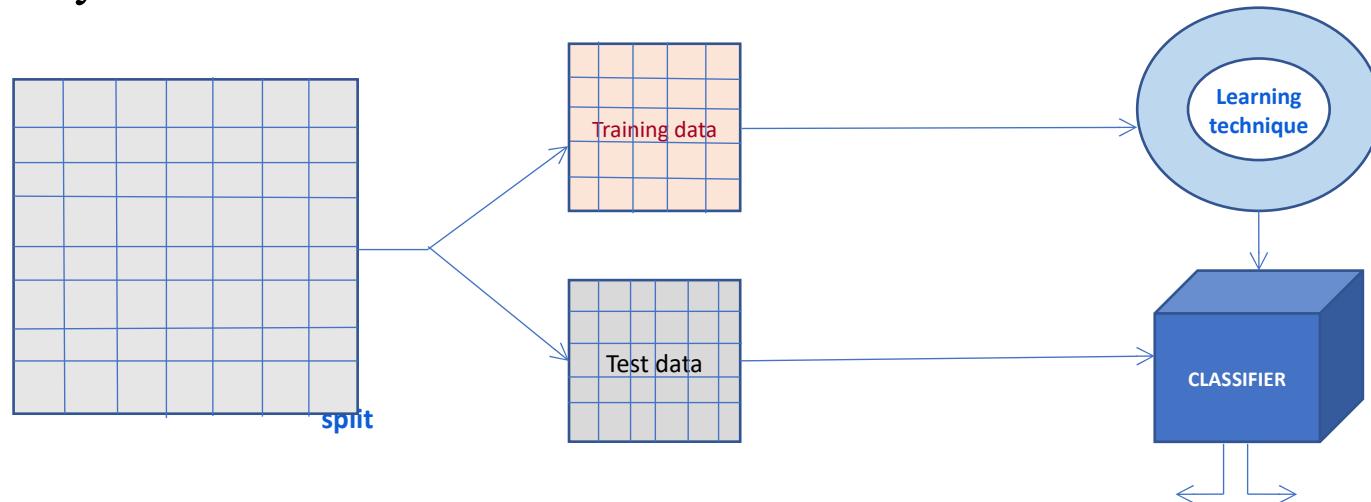
Introduction

- A classifier is used to predict an outcome of a test data
 - Such a prediction is useful in many applications
 - Business forecasting, cause-and-effect analysis, etc.
 - A number of classifiers have been evolved to support the activities.
 - Each has their own merits and demerits
- There is a need to estimate the accuracy and performance of the classifier with respect to few controlling parameters in data sensitivity
- As a task of sensitivity analysis, we have to focus on
 - Estimation strategy
 - Metrics for measuring accuracy
 - Metrics for measuring performance

Estimation Strategy

Planning for Estimation

- Using some “**training data**”, building a classifier based on certain principle is called “**learning a classifier**”.
- After building a classifier and before using it for classification of unseen instance, we have to validate it using some “**test data**”.
- Usually training data and test data are outsourced from a large pool of data already available.



Estimation Strategies

- Accuracy and performance measurement should follow a strategy. As the topic is important, many strategies have been advocated so far. Most widely used strategies are
 - Holdout method
 - Random subsampling
 - Cross-validation
 - Bootstrap approach

Holdout Method

- This is a basic concept of estimating a prediction.
 - Given a dataset, it is partitioned into **two disjoint sets** called **training set** and **testing set**.
 - Classifier is **learned** based on the training set and get **evaluated** with testing set.
 - Proportion of training and testing sets is at the discretion of analyst; typically **1:1 or 2:1**, and there is a **trade-off between these sizes** of these two sets.
 - If the training set is **too large**, then **model may be good enough**, but **estimation may be less reliable** due to small testing set and vice-versa.

Random Subsampling

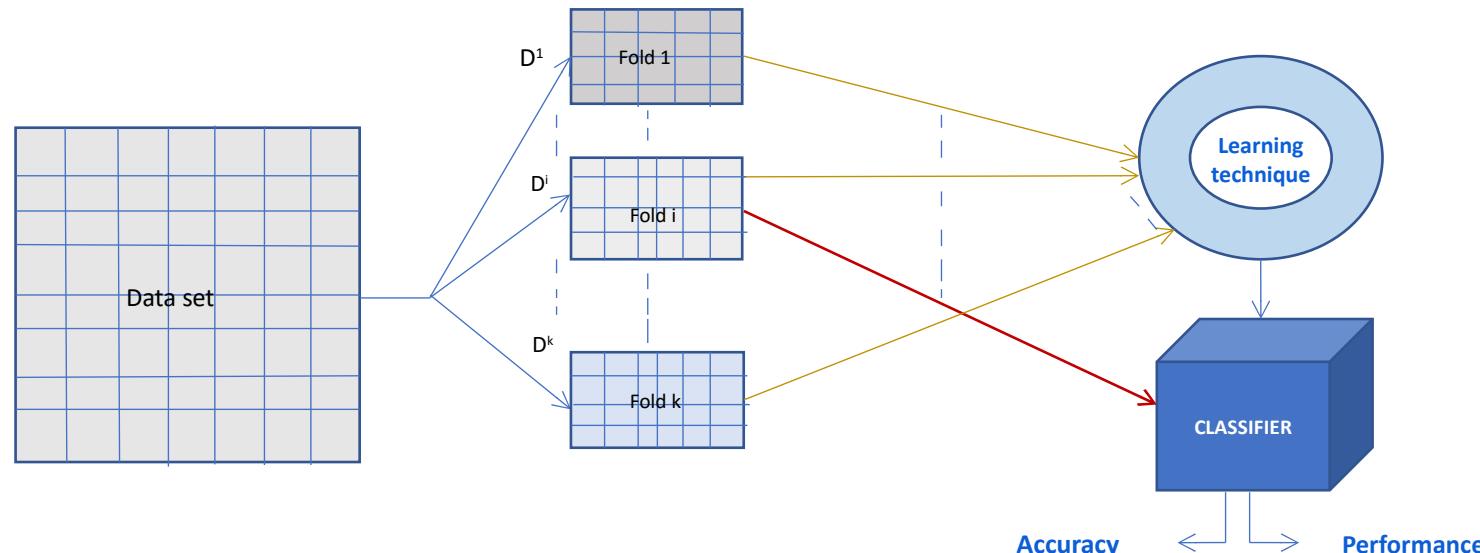
- It is a variation of Holdout method to overcome the drawback of over-presenting a class in one set thus under-presenting it in the other set and vice-versa.
- In this method, Holdout method is repeated k times, and in each time, two disjoint sets are chosen at random with a predefined sizes.
- Overall estimation is taken as the average of estimations obtained from each iteration.

Cross-Validation

- The main drawback of Random subsampling is, it does not have control over the number of times each tuple is used for training and testing.
- Cross-validation is proposed to overcome this problem.
- There are two variations in the cross-validation method.
 - k-fold cross-validation
 - N -fold cross-validation

k-fold Cross-Validation

- Dataset consisting of N tuples is divided into k (usually, 5 or 10) equal, mutually exclusive parts or folds (, and if N is not divisible by k , then the last part will have fewer tuples than other ($k-1$) parts.
- A series of k runs is carried out with this decomposition, and in i^{th} iteration is used as test data and other folds as training data
 - Thus, each tuple is used same number of times for training and once for testing.
- Overall estimate is taken as the average of estimates obtained from each iteration.



N-fold Cross-Validation

- In k -fold cross-validation method, part of the given data is used in training with k -tests.
- N -fold cross-validation is an **extreme case** of k -fold cross validation, often known as “Leave-one-out” cross-validation.
- Here, dataset is divided into as many folds as there are instances; thus, all most each tuple forming a training set, building N classifiers.
- In this method, therefore, N classifiers are built from $N-1$ instances, and each tuple is used to classify a single test instances.
- Test sets are mutually exclusive and effectively cover the entire set (in sequence). This is as if **trained by entire data as well as tested by entire data** set.
- Overall estimation is then averaged out of the results of N classifiers.

N-fold Cross-Validation : Issue

- So far the estimation of accuracy and performance of a classifier model is concerned, the *N*-fold cross-validation is comparable to the others we have just discussed.
- The drawback of *N*-fold cross validation strategy is that it is computationally expensive, as here we have to repeat the run *N* times; this is particularly true when data set is large.
- In practice, the method is extremely beneficial with very small data set only, where as much data as possible to need to be used to train a classifier.

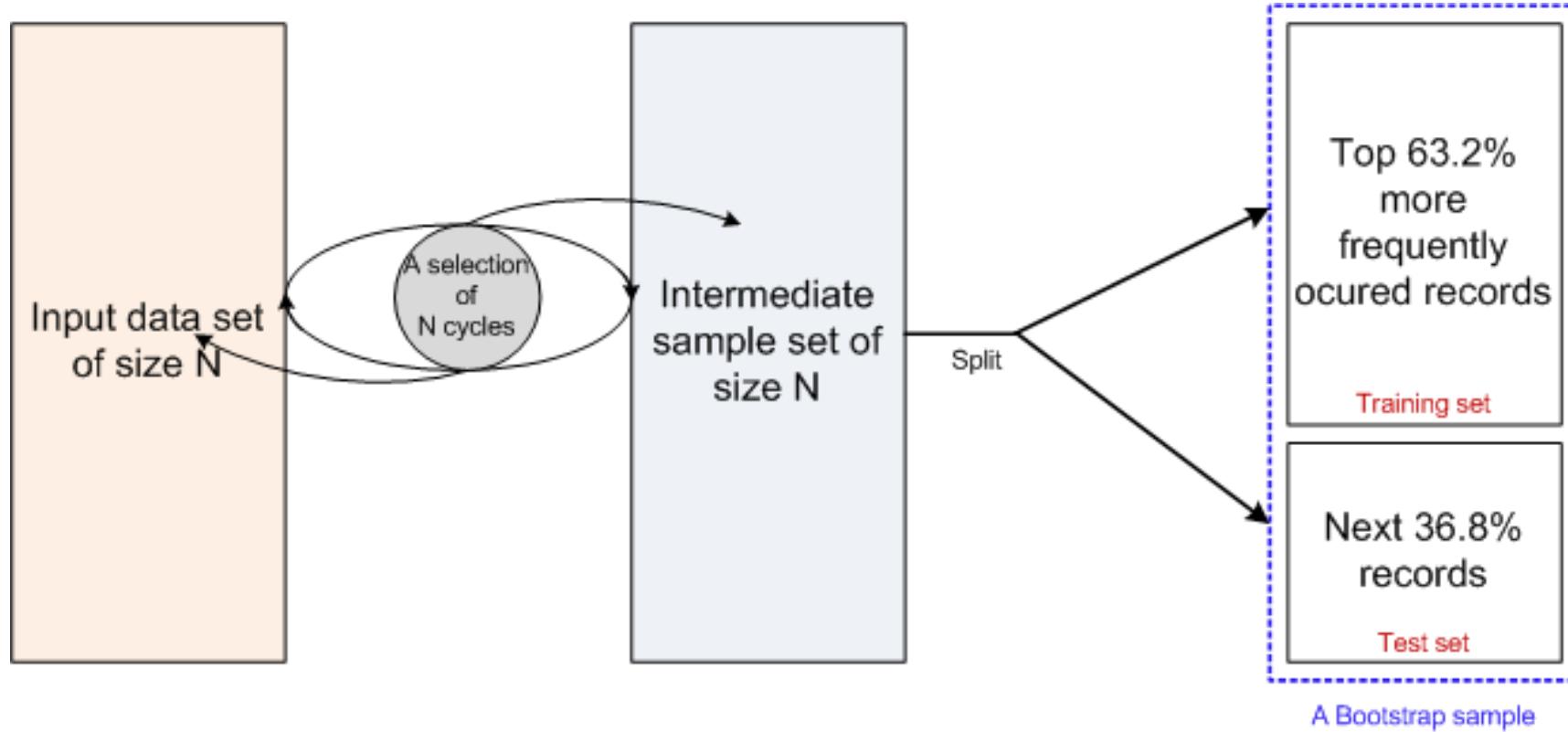
Bootstrap Method

- The Bootstrap method is a variation of **repeated version of Random sampling** method.
- The method suggests the **sampling of training records with replacement**.
 - Each time a record is selected for training set, is put back into the original pool of records, so that it is equally likely to be redrawn in the next run.
 - In other words, the Bootstrap method samples the given data set **uniformly with replacement**.
- The rational of having this strategy is that let some records be occur **more than once** in the samples of both training as well as testing.
 - What is the probability that a record will be selected more than once?

Bootstrap Method

- Suppose, we have given a data set of N records. The data set is sampled N times with replacement, resulting in a bootstrap sample (i.e., training set) of I samples.
 - Note that the entire runs are called a bootstrap sample in this method.
- There are certain chance (i.e., probability) that a particular tuple occurs **one or more** times in the training set
 - If they do not appear in the training set, then they will end up in the test set.
 - Each tuple has a probability of being selected (and the probability of not being selected is .
 - We have to select N times, so the probability that a record will not be chosen during the whole run is
 - Thus, the probability that a record is chosen by a bootstrap sample is
 - For a large value of N , it can be proved that
 - record chosen in a bootstrap sample is = 0.632

Bootstrap Method : Implication



- This is why, the Bootstrap method is also known as 0.632 bootstrap method

Accuracy Estimation

Accuracy Estimation

- We have learned how a classifier system can be tested. Next, we are to learn the metrics with which a classifier should be estimated.
- There are mainly two things to be measured for a given classifier
 - Accuracy
 - Performance
- **Accuracy estimation**
 - If N is the number of instances with which a classifier is tested and p is the number of correctly classified instances, the accuracy can be denoted as
 - Also, we can say the **error rate** (i.e., misclassification rate) denoted by ϵ is denoted by
 -

Accuracy : True and Predictive

- Now, this accuracy may be **true** (or absolute) accuracy or **predicted** (or optimistic) accuracy.
- **True accuracy** of a classifier is the accuracy when the classifier is tested with **all possible unseen instances** in the given classification space.
 - However, the number of possible unseen instances is potentially very large (if it is not infinite)
 - For example, classifying a hand-written character
 - Hence, measuring the true accuracy beyond the dispute is impractical.
- **Predictive accuracy** of a classifier is an **accuracy estimation** for a given **test data** (which are mutually exclusive with training data).
 - If the predictive accuracy for test set is α and if we test the classifier with a different test set it is very likely that a different accuracy would be obtained.
 - The predictive accuracy when estimated with a given test set it should be acceptable without any objection

Performance Estimation

Performance Estimation of a Classifier

- Predictive accuracy works fine, when the [classes are balanced](#)
 - That is, every class in the data set are equally important
- In fact, data sets with imbalanced class distributions are quite common in many real life applications
- When the classifier classified a test data set with imbalanced class distributions then, predictive accuracy on its own is not a reliable indicator of a classifier's effectiveness.

[Example 22.1: Effectiveness of Predictive Accuracy](#)

- Given a data set of stock markets, we are to classify them as “good” and “worst”. Suppose, in the data set, out of 100 entries, 98 belong to “good” class and only 2 are in “worst” class.
 - With this data set, if classifier’s predictive accuracy is 0.98, a very high value!
 - Here, there is a high chance that 2 “worst” stock markets may incorrectly be classified as “good”
 - On the other hand, if the predictive accuracy is 0.02, then none of the stock markets may be classified as “good”

Performance Estimation of a Classifier

- Thus, when the classifier classified a test data set with imbalanced class distributions, then predictive accuracy on its own is not a reliable indicator of a classifier's effectiveness.
- This necessitates an alternative metrics to judge the classifier.
- Before exploring them, we introduce the concept of **Confusion matrix**.

Confusion Matrix

- A confusion matrix for a two classes (+, -) is shown below.

	C ₁	C ₂
C ₁	True positive	False negative
C ₂	False positive	True negative

	+	-
+	++	+-
-	-+	--

- There are four quadrants in the confusion matrix, which are symbolized as below.
 - True Positive** (TP: f_{++}) : The number of instances that were positive (+) and correctly classified as positive (+v).
 - False Negative** (FN: f_{+-}): The number of instances that were positive (+) and incorrectly classified as negative (-).
 - False Positive** (FP: f_{-+}): The number of instances that were negative (-) and incorrectly classified as (+).
 - True Negative** (TN: f_{--}): The number of instances that were negative (-) and correctly classified as (-).

Confusion Matrix

Example 22.2: Confusion matrix

A classifier is built on a dataset regarding Good and Worst classes of stock markets. The model is then tested with a test set of 10000 unseen instances. The result is shown in the form of a confusion matrix. The result is self explanatory.

Class	Good	Worst	Total
Good	6954	46	7000
Worst	412	2588	3000
Total	7366	2634	10000

Predictive accuracy?

Performance Evaluation Metrics

- We now define a number of metrics for the measurement of a classifier.
 - In our discussion, we shall make the assumptions that there are only two classes: + (positive) and – (negative)
 - Nevertheless, the metrics can easily be extended to multi-class classifiers (with some modifications)
- **True Positive Rate (TPR):** It is defined as the fraction of the positive examples predicted correctly by the classifier.

=

- This metric is also known as *Recall*, *Sensitivity* or *Hit rate*.
- **False Positive Rate (FPR):** It is defined as the fraction of negative examples classified as positive class by the classifier.
- This metric is also known as *False Alarm Rate*.

Performance Evaluation Metrics

- **False Negative Rate (FNR):** It is defined as the fraction of positive examples classified as a negative class by the classifier.
- **True Negative Rate (TNR):** It is defined as the fraction of negative examples classified correctly by the classifier
 - This metric is also known as *Specificity*.

Performance Evaluation Metrics

- **Positive Predictive Value (PPV):** It is defined as the fraction of the positive examples classified as positive that are really positive
 - It is also known as *Precision*.
- **F₁ Score (F₁):** Recall (r) and Precision (p) are two widely used metrics employed in analysis, where detection of one of the classes is considered more significant than the others.
 - It is defined in terms of (r or TPR) and (p or PPV) as follows.

Predictive Accuracy (ε)

- It is defined as the fraction of the number of examples that are correctly classified by the classifier to the total number of instances.
- This accuracy is equivalent to F_w with $w_1 = w_2 = w_3 = w_4 = 1$.

Error Rate ()

- The error rate is defined as the fraction of the examples that are incorrectly classified.

Note

.

Accuracy, Sensitivity and Specificity

- Predictive accuracy () can be expressed in terms of sensitivity and specificity.
- We can write

Thus,

Analysis with Performance Measurement Metrics

- Based on the various performance metrics, we can characterize a classifier.
- We do it in terms of TPR, FPR, Precision and Recall and Accuracy
- **Case 1: Perfect Classifier**

When every instance is correctly classified, it is called the perfect classifier. In this case, $TP = P$, $TN = N$ and CM is

$$TPR = 1$$

$$FPR = 0$$

$$Precision = 1$$

$$F_1 Score = 1$$

$$Accuracy = 1$$

		Predicted Class	
		+	-
Actual class	+	P	0
	-	0	N

Analysis with Performance Measurement Metrics

- **Case 2: Worst Classifier**

When every instance is **wrongly** classified, it is called the **worst classifier**. In this case, $TP = 0$, $TN = 0$ and the CM is

$$TPR = 0$$

$$FPR = 1$$

$$Precision = 0$$

F_1 Score = Not applicable
as $Recall + Precision = 0$

$$\text{Accuracy} = 0$$

		Predicted Class	
		+	-
Actual class	+	0	P
	-	N	0

Analysis with Performance Measurement Metrics

- **Case 3: Ultra-Liberal Classifier**

The classifier always predicts the + class correctly. Here, the False Negative (FN) and True Negative (TN) are zero. The CM is

$$TPR = 1$$

$$FPR = 0$$

$$Precision =$$

$$F_1 Score =$$

$$\text{Accuracy} =$$

		Predicted Class	
		+	-
Actual class	+	P	0
	-	N	0

Analysis with Performance Measurement Metrics

- **Case 4: Ultra-Conservative Classifier**

This classifier always predicts the - class correctly. Here, the False Negative (FN) and True Negative (TN) are zero. The CM is

$$\boxed{\begin{aligned} TPR &= = 0 \\ FPR &= = 0 \\ Precision &= \\ &\quad \text{(as } TP + FP = 0\text{)} \\ F_1 Score &= \\ \text{Accuracy} &= \end{aligned}}$$

		Predicted Class	
		+	-
Actual class	+	0	p
	-	0	N

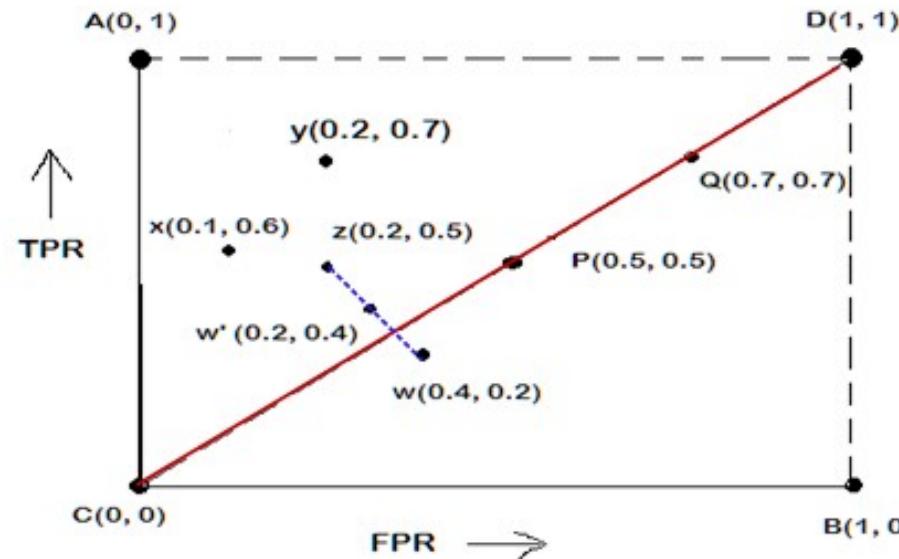
ROC Curves

ROC Curves

- ROC is an abbreviation of **Receiver Operating Characteristic** come from the signal detection theory, developed during World War 2 for analysis of radar images.
- In the context of classifier, ROC plot is a useful tool to study the behaviour of a classifier or **comparing two or more classifiers**.
- A ROC plot is **a two-dimensional graph**, where, X-axis represents FP rate (FPR) and Y-axis represents TP rate (TPR).
- Since, the values of FPR and TPR varies from 0 to 1 both inclusive, the two axes thus from 0 to 1 only.
- Each point (x, y) on the plot indicating that the FPR has value x and the TPR value y .

Interpretation of Different Points in ROC Plot

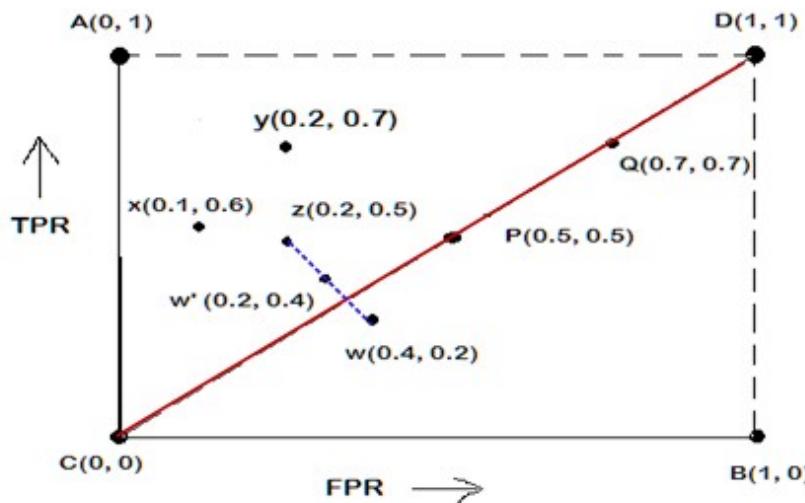
- Let us interpret the different points in the ROC plot.



- The four points (A, B, C, and D)
 - A: TPR = 1, FPR = 0, the ideal model, i.e., the **perfect classifier**, no false results
 - B: TPR = 0, FPR = 1, the **worst classifier**, not able to predict a single instance
 - C: TPR = 0, FPR = 0, the model predicts every instance to be a **Negative class**, i.e., it is an **ultra-conservative classifier**
 - D: TPR = 1, FPR = 1, the model predicts every instance to be a **Positive class**, i.e., it is an **ultra-liberal classifier**

Interpretation of Different Points in ROC Plot

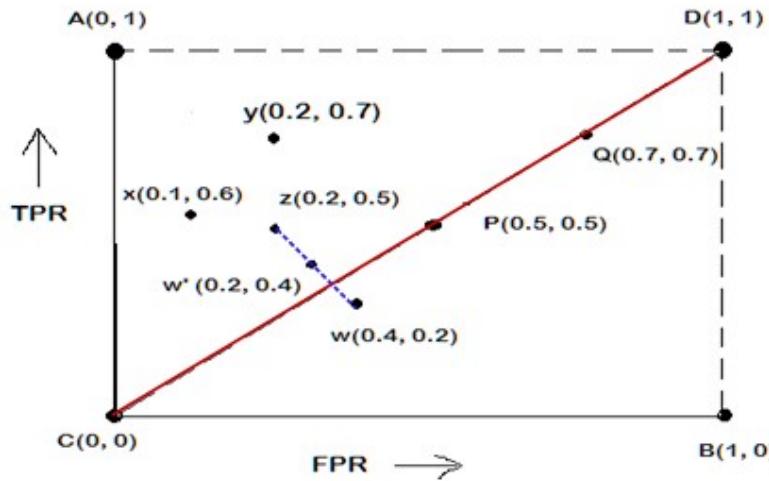
- Let us interpret the different points in the ROC plot.



- The points on diagonals
 - The diagonal line joining point C(0,0) and D(1,1) corresponds to random guessing
 - Random guessing means that a record is classified as positive (or negative) with a certain probability
 - Suppose, a test set containing N_+ positive and N_- negative instances. Suppose, the classifier guesses any instances with probability p
 - Thus, the random classifier is expected to correctly classify $p.N_+$ of the positive instances and $p.N_-$ of the negative instances
 - Hence, $TPR = FPR = p$
 - Since $TPR = FPR$, the random classifier results reside on the main diagonals

Interpretation of Different Points in ROC Plot

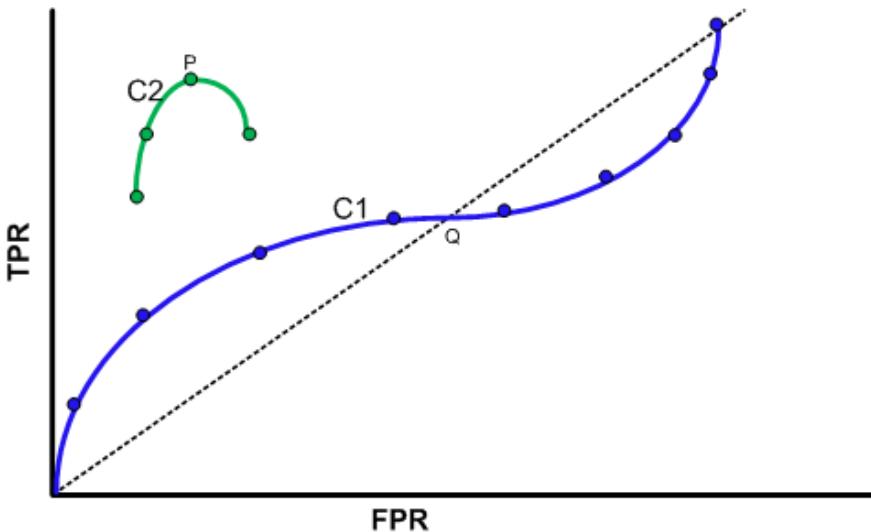
- Let us interpret the different points in the ROC plot.



- The points on the upper diagonal region
 - All points, which reside on upper-diagonal region are corresponding to classifiers “good” as their TPR is as good as FPR (i.e., FPRs are lower than TPRs)
 - Here, X is better than Z as X has higher TPR and lower FPR than Z.
 - If we compare X and Y, neither classifier is superior to the other

Tuning a Classifier through ROC Plot

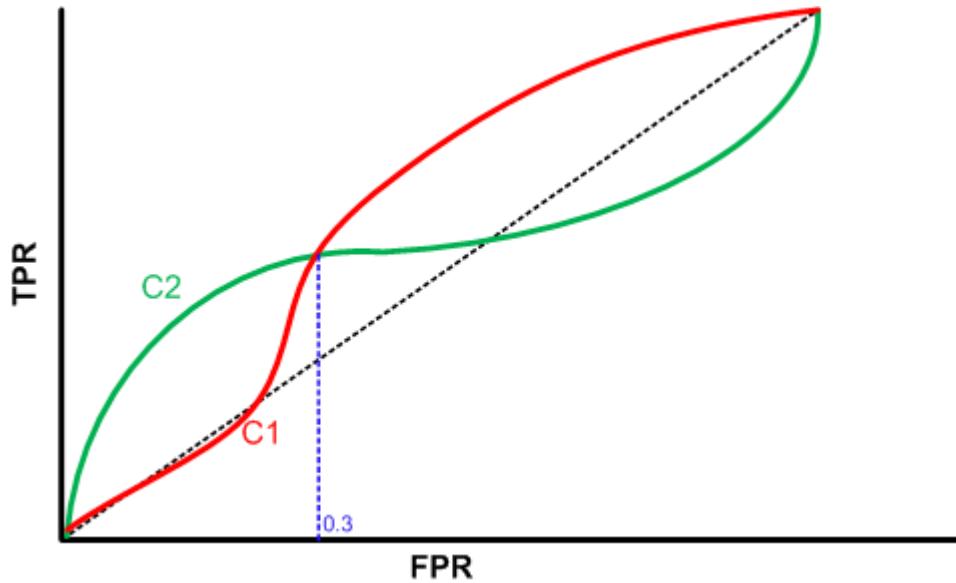
- Using ROC plot, we can compare two or more classifiers by their TPR and FPR values and this plot also depicts the trade-off between TPR and FPR of a classifier.



- Examining ROC curves can give insights into the best way of tuning parameters of classifier.
- For example, in the curve C2, the result is degraded after the point P. Similarly for the observation C1, beyond Q the settings are not acceptable.

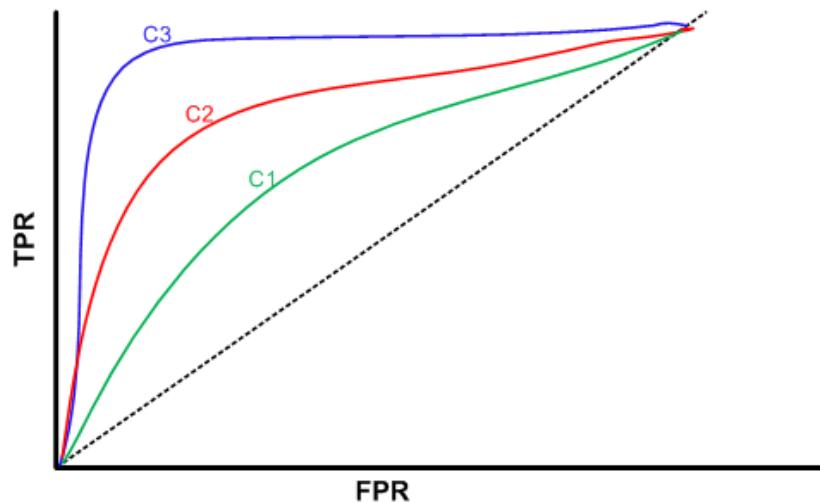
Comparing Classifiers through ROC Plot

- Two curves C1 and C2 are corresponding to the experiments to choose two classifiers with their parameters.
- Here, C1 is better than C2 when FPR is less than 0.3.
- However, C2 is better, when FPR is greater than 0.3.
- Clearly, neither of these two classifiers dominates the other.



Comparing Classifiers through ROC Plot

- We can use the concept of “**area under curve**” (AUC) as a better method to compare two or more classifiers.
- If a model is perfect, then its AUC = 1.
- If a model simply performs random guessing, then its AUC = 0.5
- A model that is strictly better than other, would have a larger value of AUC than the other.



- Here, C3 is best, and C2 is better than C1 as $AUC(C3) > AUC(C2) > AUC(C1)$.

Any question?

Signal Processing

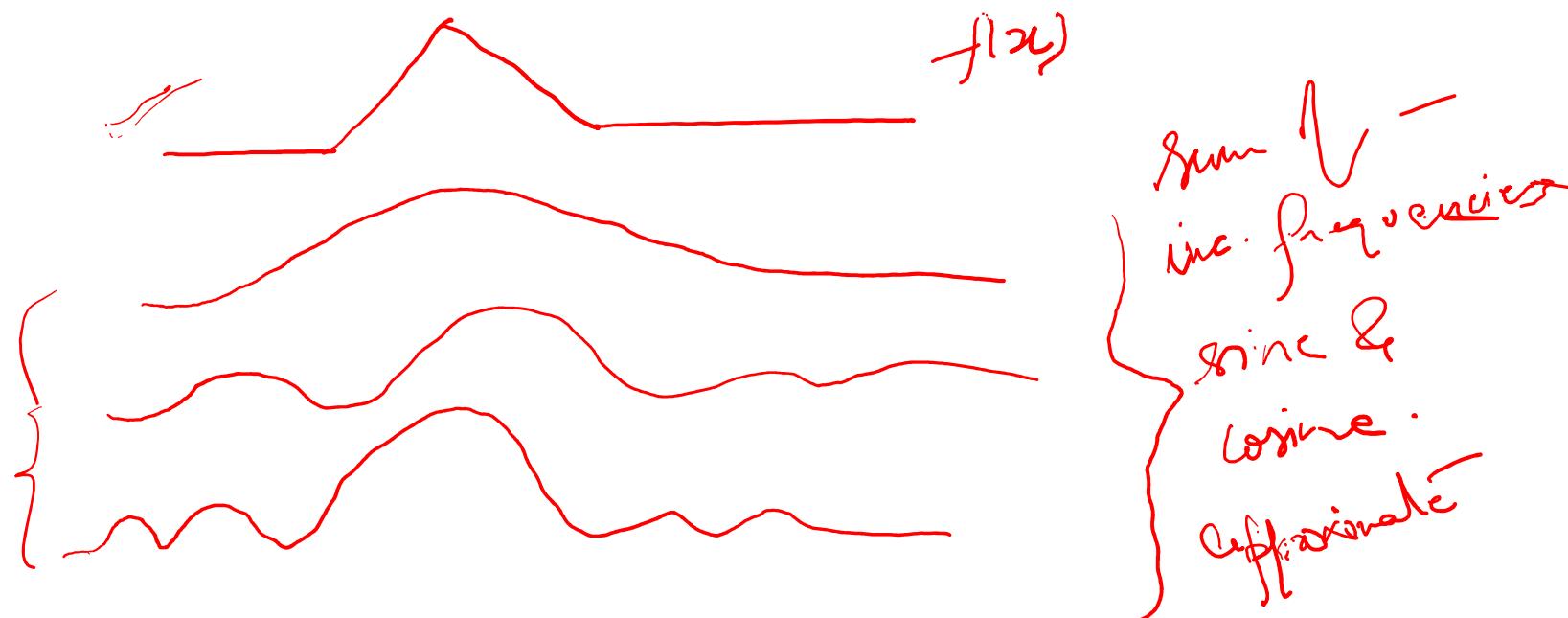
Frequency Domain Analysis

Frequency Domain Analysis

- EEG signals are recorded in the domain of **TIME**.
- Introduced by J.-B. Joseph Fourier in the early 1800s
- It's a coordinate transform system
- Fourier introduced the concept that sine and cosine functions of increasing frequency provide an orthogonal basis for the space of solution functions.

Fourier Series

- Any arbitrary function can be represented using sine and cosines.
- Sine and cosine form the basis of the function space.

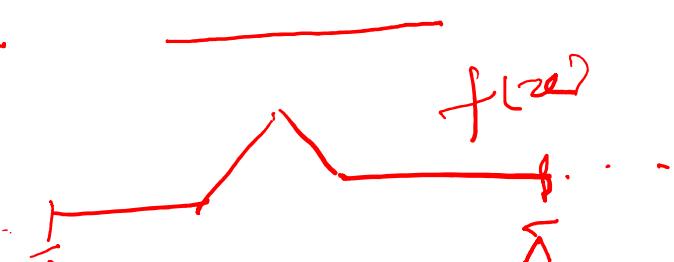


Fourier Series

- Approximating arbitrary function $f(x)$, as a infinite sum of sine and cosine of increasing high frequency.

$$f(x) = \frac{A_0}{2} + \sum_{k=1}^{\infty} [A_k \cos(kx) + B_k \sin(kx)]$$

Where A_k & B_k are Fourier constants



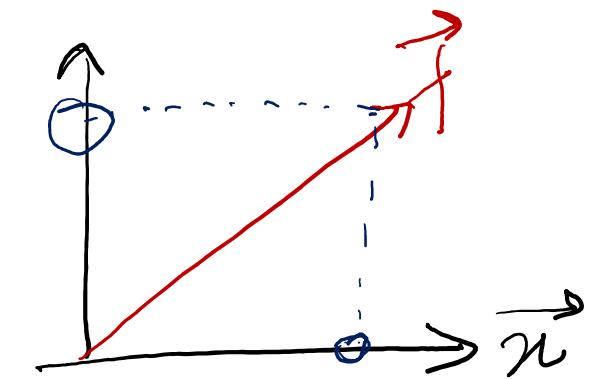
$$A_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx = \frac{1}{\|\cos(kx)\|^2} \langle f(x), \cos(kx) \rangle$$

$$B_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx = \frac{1}{\|\sin(kx)\|^2} \langle f(x), \sin(kx) \rangle$$

Fourier Series

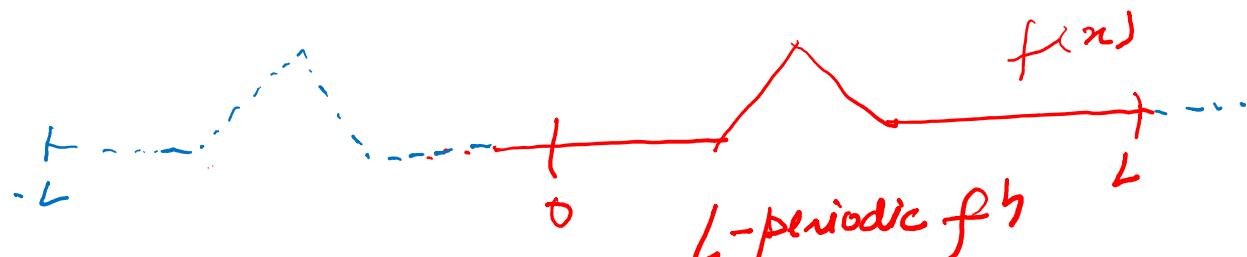
$$\vec{f} = \underbrace{\langle \vec{f}, \vec{x} \rangle}_{= \vec{y}} \cdot \frac{\vec{x}}{\|\vec{x}\|^2} + \langle \vec{f}, \vec{y} \rangle \cdot \frac{\vec{y}}{\|\vec{y}\|^2}$$

$$f(x) = \underbrace{f(x) \cdot \cos(kx)}_{\|f\|_1} \cdot \frac{\cos kx}{\|\cos kx\|^2} + \text{flr. sin(kx)} \frac{\sin kx}{\|\sin kx\|^2}$$



$\rightarrow f^n$ space

Fourier Series



$$f(x) = \frac{A_0}{2} + \sum_{k=1}^{\infty} \left(A_k \cos\left(\frac{2\pi k x}{L}\right) + B_k \sin\left(\frac{2\pi k x}{L}\right) \right)$$

$$A_k = \frac{2}{L} \int_0^L f(x) \cdot \cos\left(\frac{2\pi k x}{L}\right) dx$$

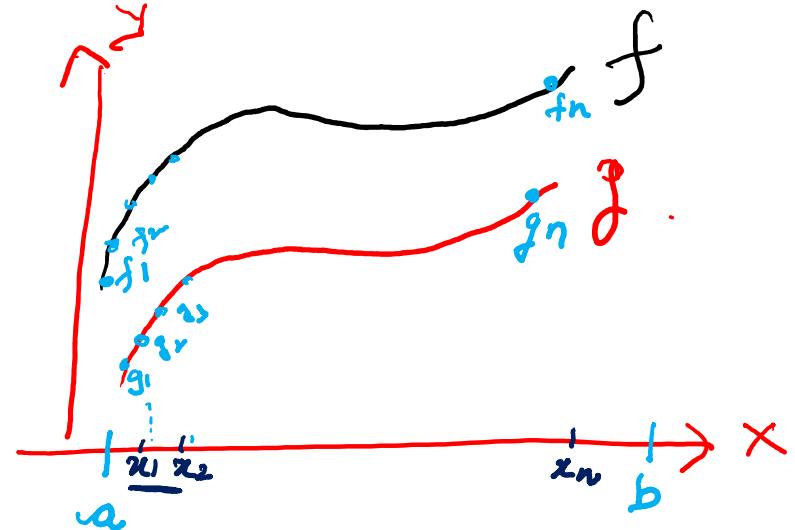
$$B_k = \frac{2}{L} \int_0^L f(x) \cdot \sin\left(\frac{2\pi k x}{L}\right) dx$$

Inner Product of Functions

$$f = [f_1 \ f_2 \ \dots \ f_n]^T$$

$$g = [g_1 \ g_2 \ \dots \ g_m]^T$$

$$\langle f(x), g(x) \rangle = \int_a^b f(x) g(x) dx$$



$$\Delta x = \frac{b-a}{n-1}$$

When 'n' data points are increasing

$$\frac{b-a}{n-1} \langle f, g \rangle = \sum_{k=1}^n f(x_k) g(x_k) \cdot \Delta x$$

$$n \rightarrow \infty \equiv \Delta x \rightarrow 0$$

Fourier Series for Complex functions

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}$$

$c^{ikx} = \cos(kx) + i \sin(kx)$
 Euler's expansion
 $i = \sqrt{-1}$

$$f(x) = \sum_{k=-\infty}^{\infty} (\alpha_k + i \beta_k) [\cos(kx) + i \sin(kx)]$$

$$= (\alpha_0 + i \beta_0) + \sum_{k=1}^{\infty} \left[(\underline{\alpha_{-k}} + \underline{\alpha_k}) \cos(kx) + (\underline{\beta_{-k}} - \underline{\beta_k}) \sin(kx) \right]$$

$$+ i \sum_{k=1}^{\infty} \left[(\beta_{-k} + \beta_k) \cos(kx) - (\alpha_{-k} - \alpha_k) \sin(kx) \right]$$

$$\text{Let } e^{ikx} = \cos(kx) + \sin(kx) = \psi_k \quad k \in \mathbb{Z}$$

$$\begin{aligned} \langle \psi_j, \psi_k \rangle &= \int_{-\pi}^{\pi} e^{ijx} e^{-ikx} dx = \int_{-\pi}^{\pi} e^{i(j-k)x} dx \\ &= \frac{1}{i(j-k)} \left[e^{i(j-k)x} \right]_{-\pi}^{\pi} = \begin{cases} 0, & \text{if } j \neq k \\ 2\pi, & \text{if } j = k \end{cases} \end{aligned}$$

We can write as

$$\frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \langle f(x), \psi_k \rangle \psi_k$$

$\underbrace{\hspace{10em}}$

ψ_k e^{ikx}

$$\|\psi_k\|^2 = 2\pi$$

③ Fourier Transform is a linear operator

$$\hat{F}(\alpha f(\omega) + \beta g(\omega)) = \alpha \hat{F}(f(\omega)) + \beta \hat{F}(g(\omega))$$

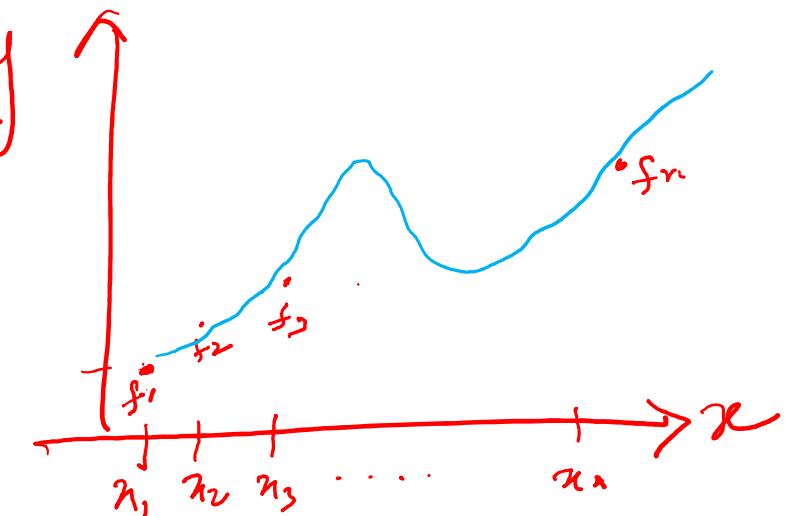
$$F^{-1}(\alpha \hat{f}(\omega) + \beta \hat{g}(\omega)) = \alpha f(\omega) + \beta g(\omega)$$

④ Parseval's theorem

$$\int_{-\infty}^{\infty} |\hat{f}(\omega)|^2 d\omega = 2\pi \int_{-\infty}^{\infty} |f(x)|^2 dx$$

Discrete Fourier Transform

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_n \end{bmatrix}$$



$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-i 2\pi j k / n}$$

$$f_k = \left[\sum_{j=0}^{n-1} \hat{f}_j e^{i 2\pi j k / n} \right] \cdot \frac{1}{n}$$

$\{f_0, f_1, f_2, \dots, f_n\} \xrightarrow{\text{DFT}} \{\hat{f}_0, \hat{f}_1, \dots, \hat{f}_n\}$
 \hat{f}_j : how much of each freq. is required to reconstruct the data in f .

$$f_j e^{-i \omega j k / n} \Rightarrow e^{-i \omega j k / n} \Rightarrow \underline{\omega_n} \rightarrow \text{fundamental freq.}$$

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-i \omega j k / n}$$

[approximation in form of
sine & cosines with
n discrete values]

$$\begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix} \xrightarrow{k} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & & & & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{e(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

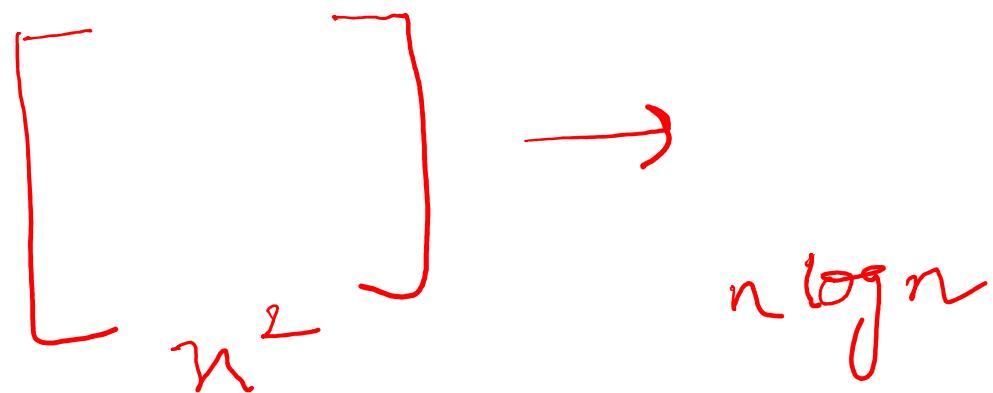
DFT \rightarrow Mathematical Transform

FFT \rightarrow computationally efficient way of computing DFT

FAST FOURIER TRANSFORM

$$\text{DFT} = \sum_{n=1}^N [] \rightarrow O(n^2)$$

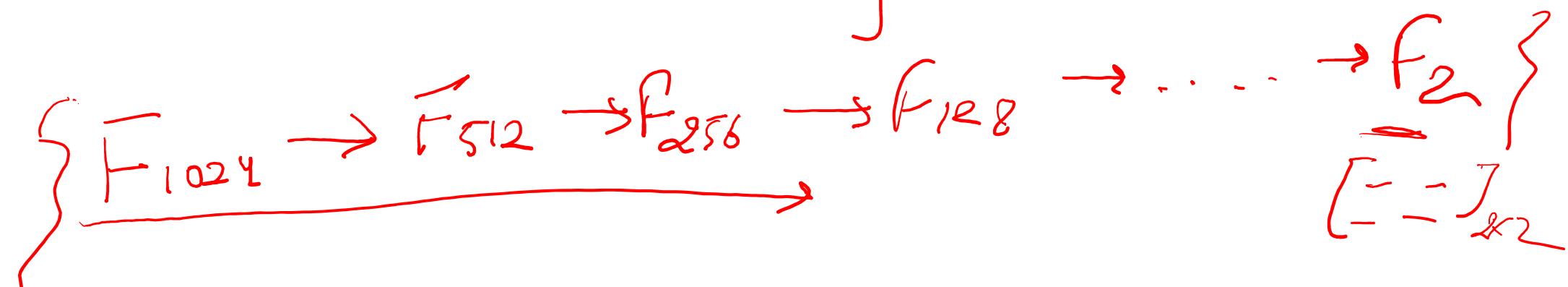
$$\text{FFT} \rightarrow O(n \log n)$$

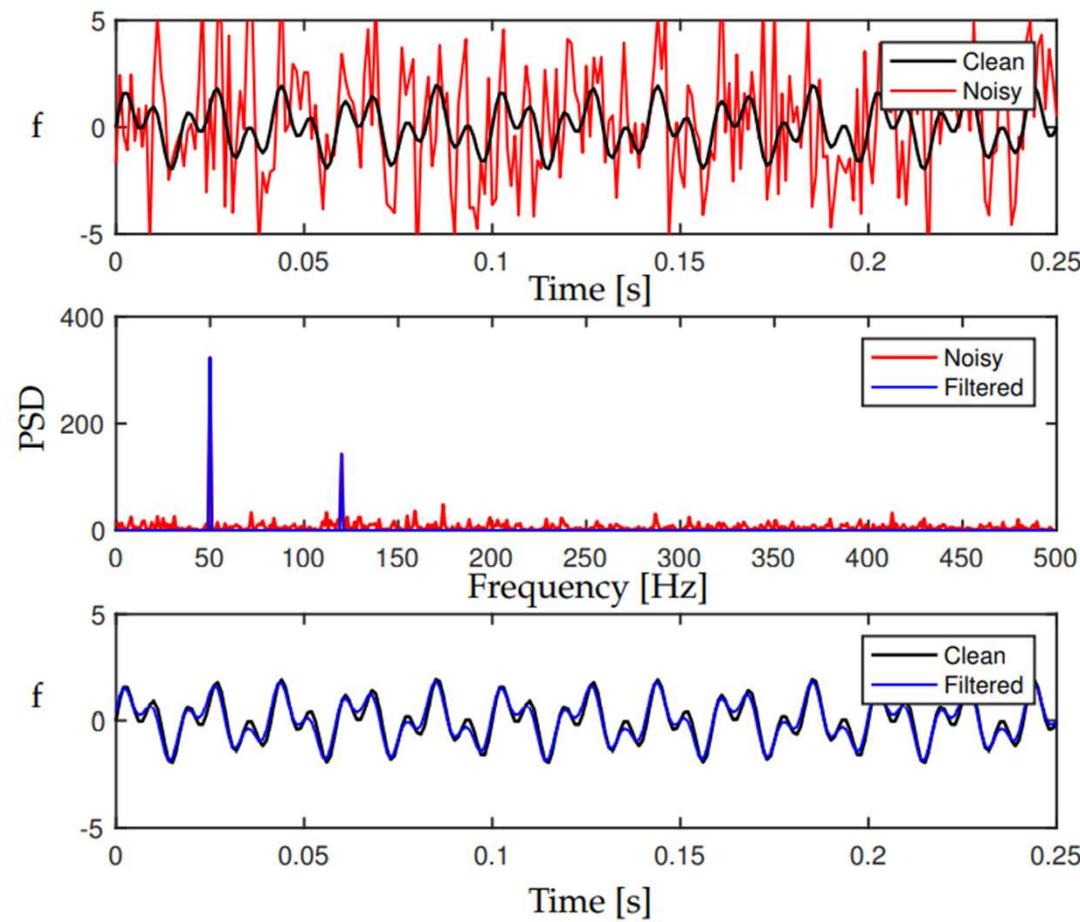


$$\hat{f} = \overline{\hat{F}_{1024}} f \quad \left\{ n = 1024 \text{ or } \underline{2^{10}} \right\}$$

$$= \frac{\begin{bmatrix} I_{512} & -D_{512} \\ I_{512} & -D_{512} \end{bmatrix} \begin{bmatrix} \hat{F}_{512} & 0 \\ 0 & \hat{F}_{512} \end{bmatrix} \begin{bmatrix} \text{for even} \\ \text{for odd} \end{bmatrix}}{\dots} \quad \left\{ \begin{array}{l} t_1 \\ f_1 \\ t_3 \\ f_3 \\ \vdots \\ t_m \\ f_m \end{array} \right\}$$

$$\underline{D_{512}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & \ddots \\ 0 & 0 & 0 & \omega^{512} \end{bmatrix}$$





Wavelet Transform

- FFT \rightarrow basis $f^n \rightarrow$ cosine & sine
 - Signals which are non-periodic, finite, discontinuous
EEG \rightarrow non-stationary
- ① \rightarrow Perform Fourier analysis over short time windows
known as Short time Fourier transform (\rightarrow STFTs)



Wavelet transform

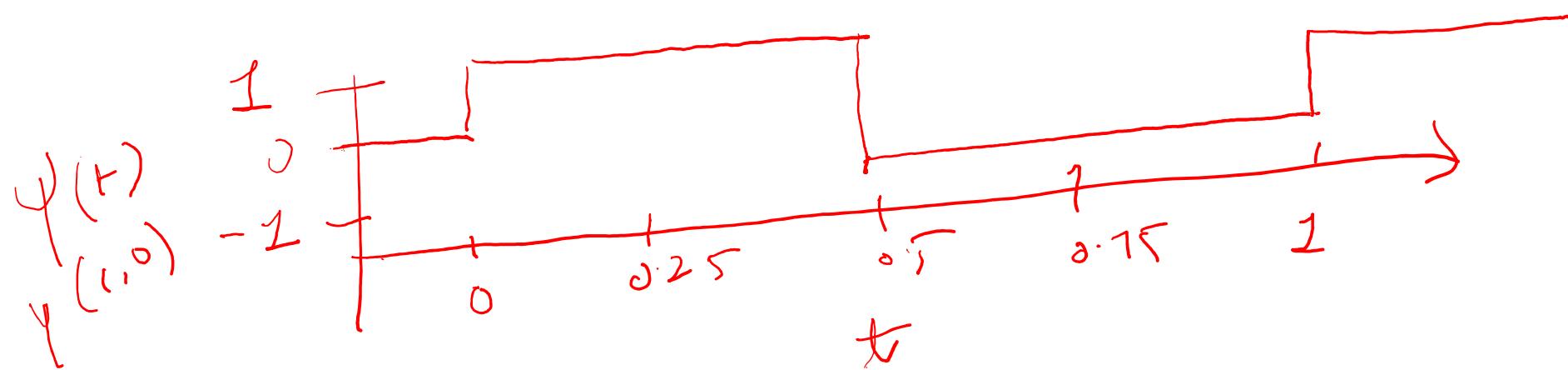
finite Basis function \rightarrow wavelets \rightarrow scaled over a
single finite length waveform \rightarrow Mother Wavelets

$\psi(t) \rightarrow$ mother wavelet

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right)$$

Haar wavelet

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & -\frac{1}{2} \leq t \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



continuous WT

$$\underline{\mathcal{W}_\psi(f)(a,b)} = \langle f, \Psi_{a,b} \rangle = \int_{-\infty}^{\infty} f(t) \underline{\Psi_{a,b}(t)} dt$$

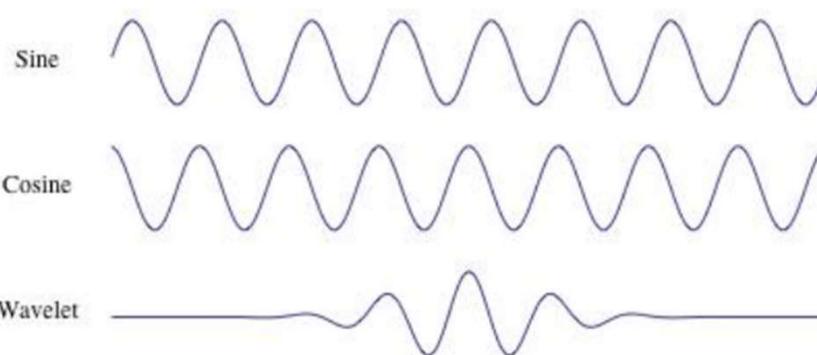
Inverse CWT

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{W}_\psi(f)(a,b) \underline{\Psi_{a,b}(t)} \frac{da db}{a^2}$$

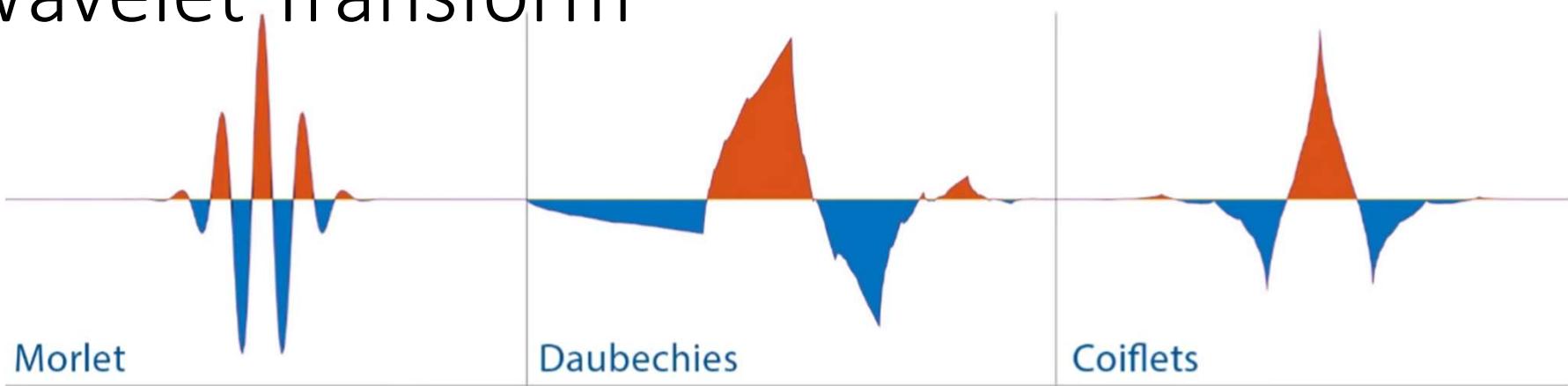
$$C_\psi = \int_{-\infty}^{\infty} \frac{|\psi(\omega)|^2}{|\omega|} d\omega$$

$$\underbrace{\mathcal{D}\omega^T}_{\mathcal{D}\psi(f)}(j,k) = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(t) \psi_{j,k}(t) dt$$

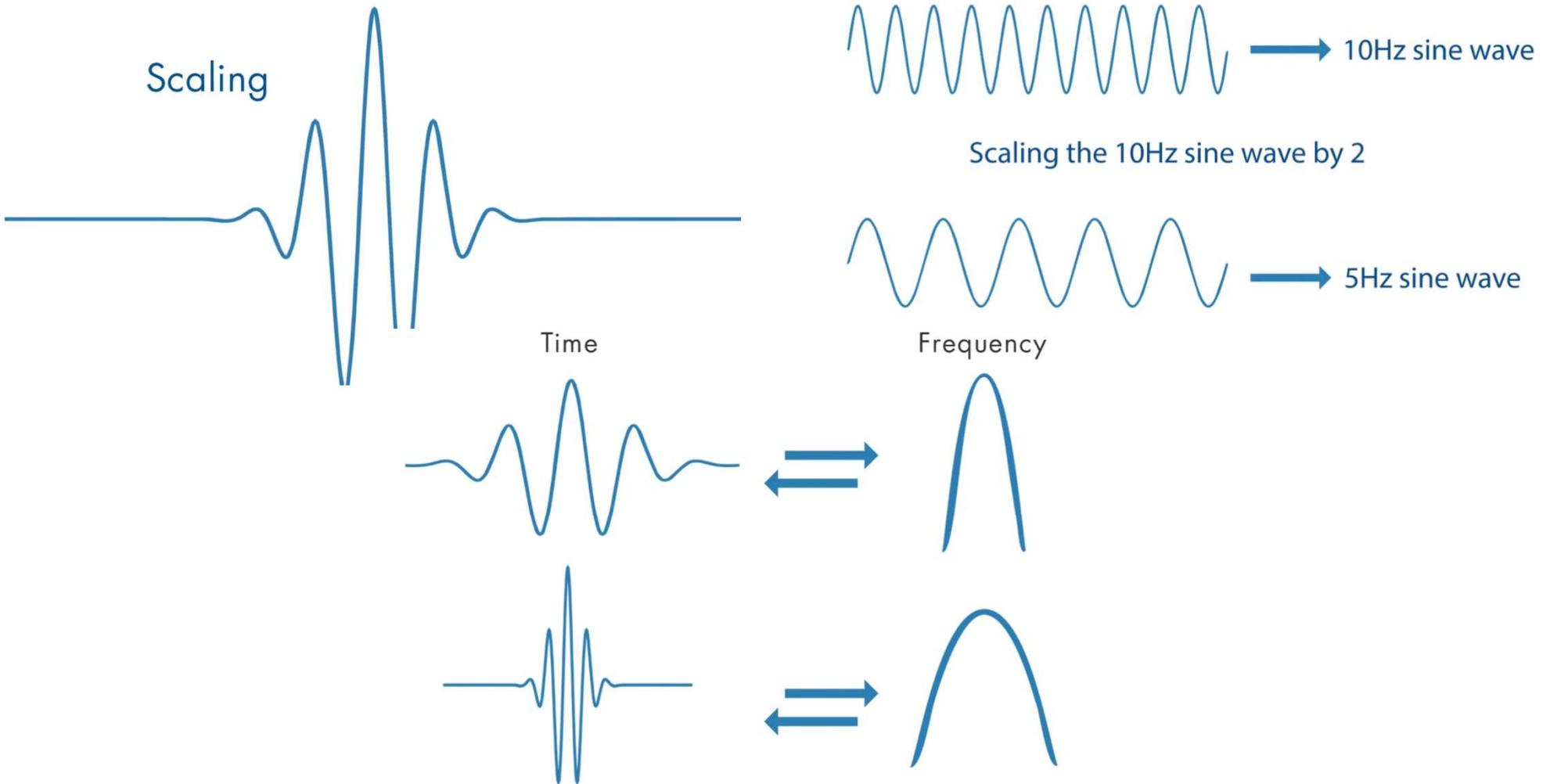
$$\psi_{j,k}(t) = \frac{1}{a_j^0} \psi\left(\frac{\theta - k\beta}{a_j^0}\right)$$



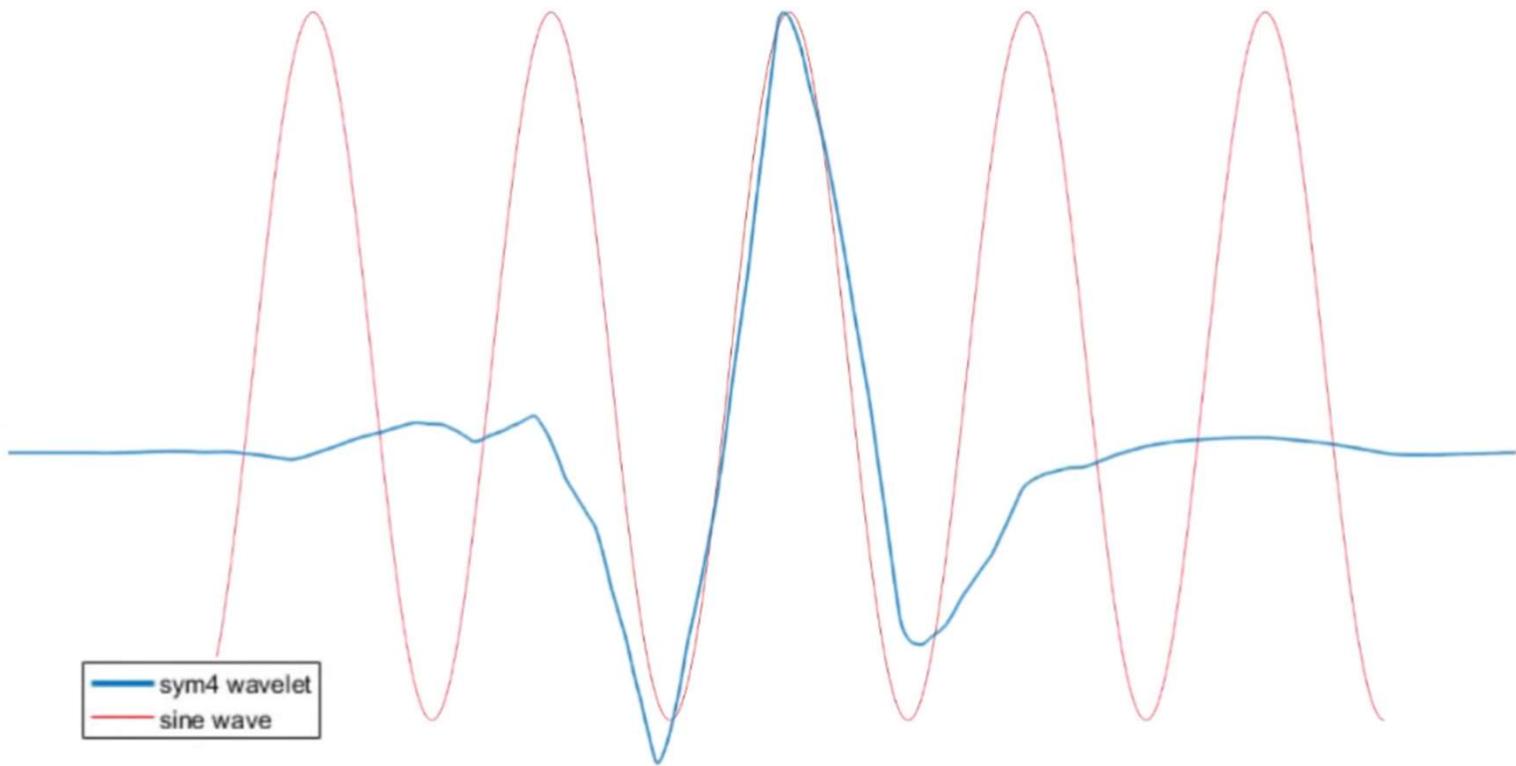
Wavelet Transform



Wavelet Transform



Wavelet Transform

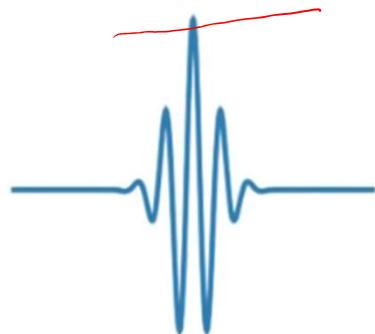


Comparison of a Wavelet with a Sine Wave of Same Frequency

Wavelet Transform



$$s > 1$$



$$0 < s < 1$$

$$\Psi\left(\frac{t}{s}\right)$$



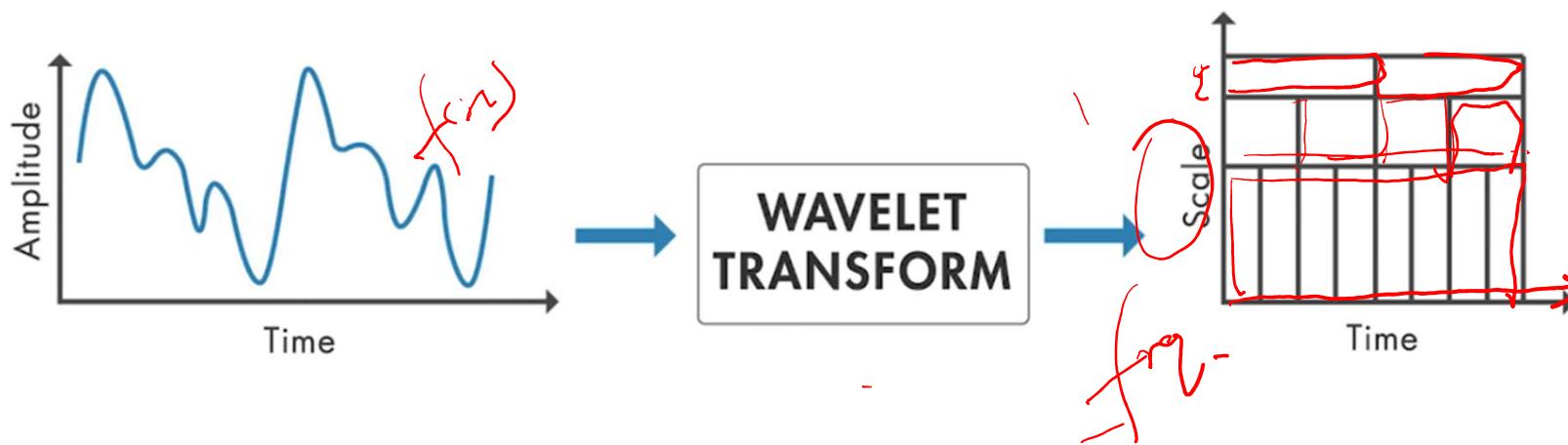
large scale factor
low frequency



small scale factor
high frequency

$$\Psi(t) \cdot \frac{1}{\sqrt{a}}$$

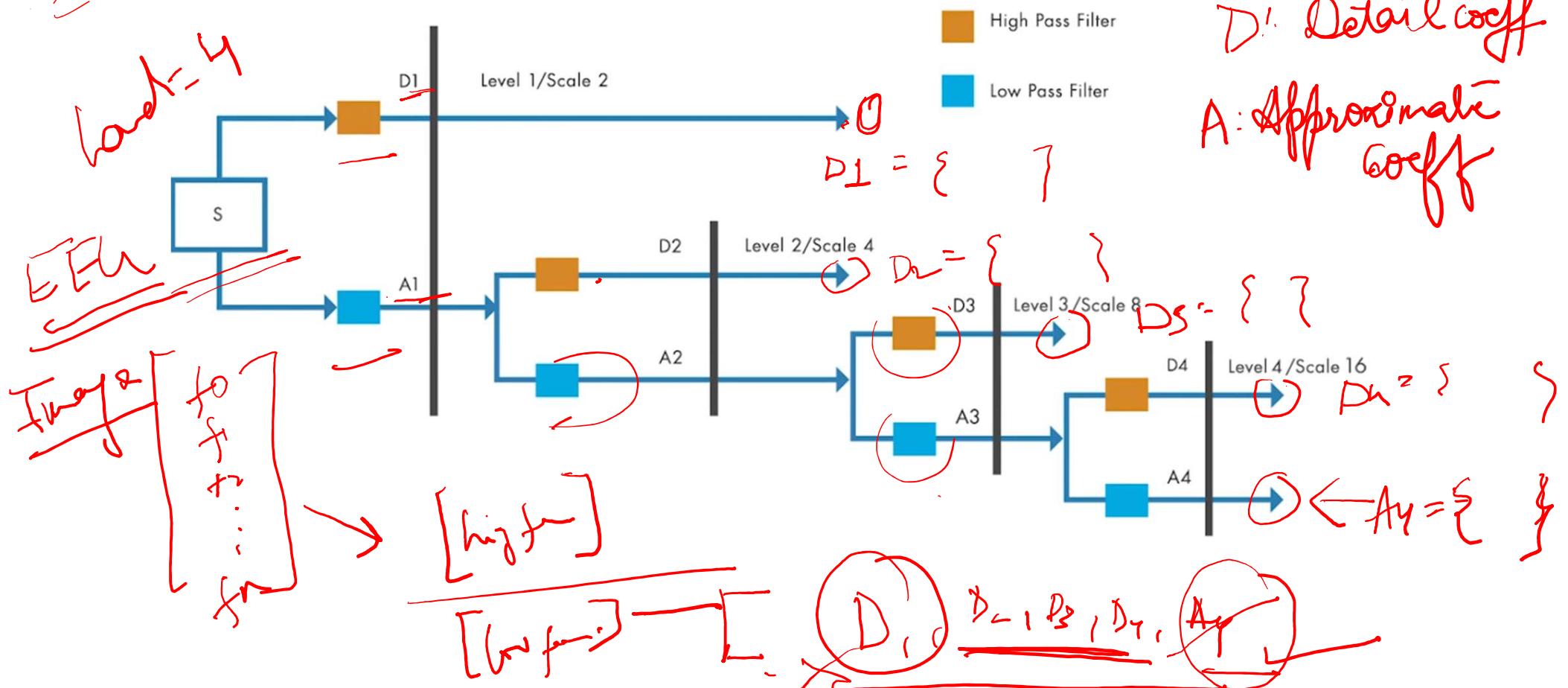
Continuous Wavelet Transform

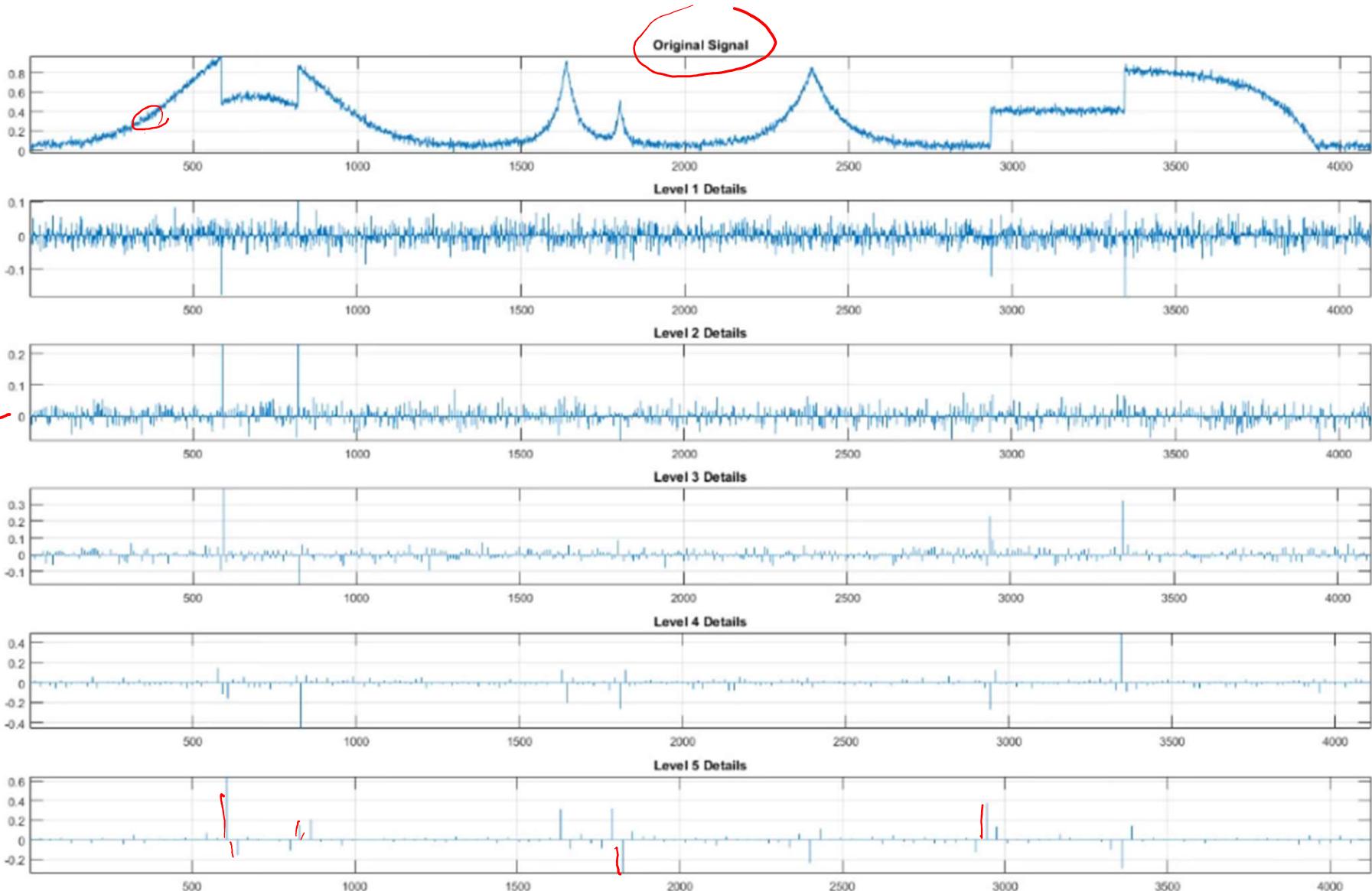


Discrete Wavelet Transform

← EEG

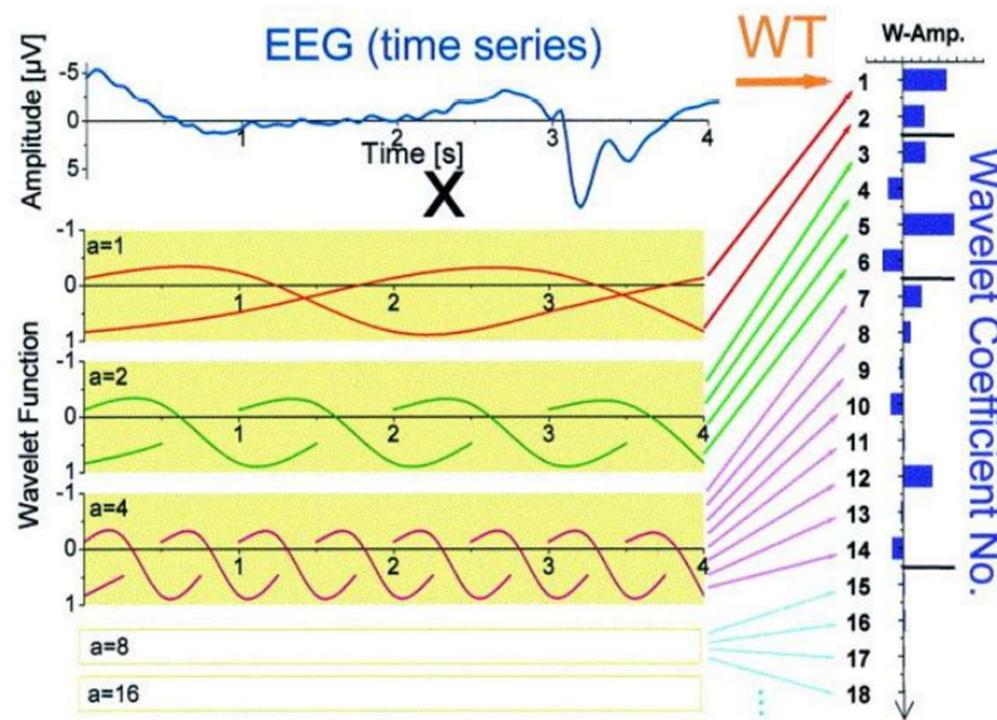
FFT / DWT





DWT
Levels

Example: Wavelet Decomposition of EEG



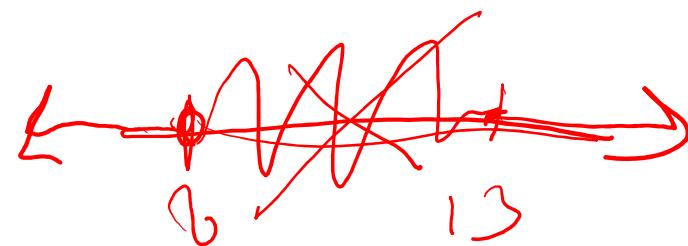
Credits: (Hinterberger et al., 2003) ↗

Filters

- High Pass Filter:
 - Blocks dc offset in high gain amplifiers or single supply circuits. Filters can be used to separate signals, passing those of interest, and attenuating the unwanted frequencies.
- Low Pass Filter:
 - Stabilizes amplifiers by rolling off the gain at higher frequencies where excessive phase shift may cause oscillations

Filters

- Band Pass Filter:
 - If a high-pass filter and a low-pass filter are cascaded, a band pass filter is created. The band pass filter passes a band of frequencies between a lower cutoff frequency, f_L , and an upper cutoff frequency, f_h
- Notch (Band Reject Filter):
 - The pass bands include frequencies below f_L and above f_h . The band from f_L to f_h is in the stop band.



TIME DOMAIN ANALYSIS

① Hjorth Parameters [1970s] ← EEG $s(t)$

↳ Mean Power

↳ RMS freq.

↳ RMS freq. spread

② Activity

$$A = a_0$$

a_0 : variance of signal in
the epoch under measure
-ment

③ Mobility

$$M = \sqrt{\frac{a_2}{a_0}}$$

$$a_2 = \text{variance of } \frac{ds(t)}{dt}$$

④ Complexity

$$C = \sqrt{\frac{a_4}{a_0}}$$

$$a_4: \text{variance of } \frac{d^2 s(t)}{dt^2}$$

② AutoRegressive Modeling (AR)

$$x_t = \sum_{i=1}^p a_i x_{t-i} + \epsilon_t$$

ϵ_t - some white noise

p - order of AR model

Adaptive AutoRegressive-(AAR) Model.

$$x_t = \sum_{i=1}^p \underbrace{a_{i,t}}_{\uparrow \text{Statistical Structure over time}} x_{t-i} + \epsilon_t$$

Spatial Filtering

Spatial Filtering

- Spatial filtering techniques take as input brain signals recorded from several different locations (or “channels”) and transform them in one of several ways.
- Possible goals include
 - enhancing local activity
 - reducing noise that is common across channels,
 - decreasing the dimensionality of the data,
 - finding projections that maximize discrimination between different classes

Bipolar

- Extract bipolar signals

$$\widetilde{S_{i,j}} = S_i - S_j$$

- Highlight the **electrical potential differences** between the two electrodes of interest (i and j).

Laplacian

- *Laplacian filtering*, extracts local activity at electrode i by subtracting the average activity present in the four orthogonal nearest neighboring electrodes

$$\tilde{s} = s_i - \frac{1}{4} \sum_{i \in \theta} s_i$$

Common Average Referencing

- *Common average referencing* (CAR), enhances the local activity at electrode i by subtracting the average over all electrodes

$$\tilde{s}_i = s_i - \frac{1}{N} \sum_{i=1}^N s_i$$

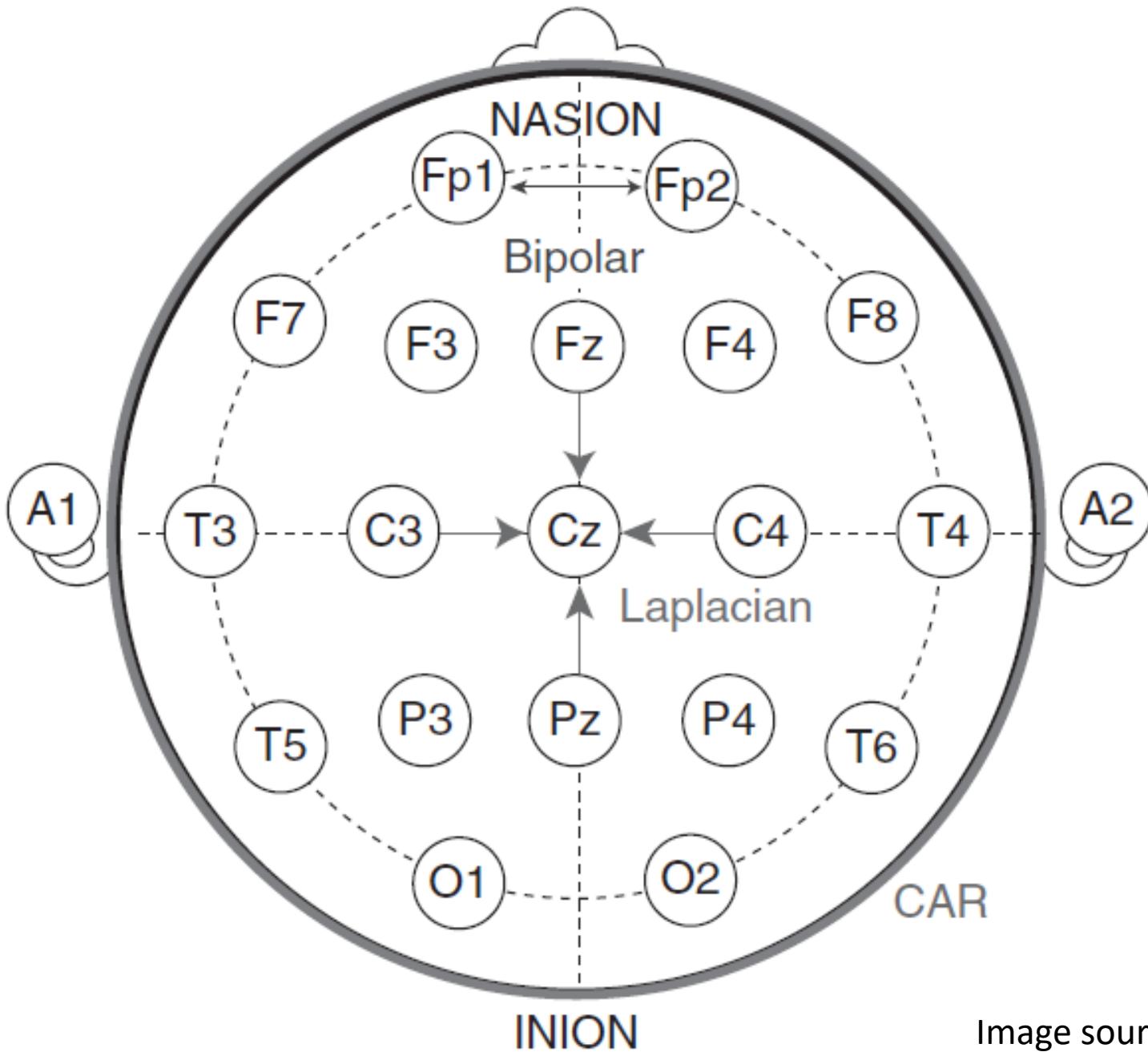


Image source: Rajesh P.N, Rao- Brain Computer Interfacing: An Introduction

Vector Representation

- A vector $\mathbf{x} \in \mathbb{R}^n$ can be represented by **n** components:
- Assuming the standard base $\langle \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N \rangle$ (i.e., unit vectors in each dimension), x_i can be obtained by **projecting** \mathbf{x} along the direction of \mathbf{v}_i :
- \mathbf{x} can be “**reconstructed**” from its projections as follows:
- Since the basis vectors are the same for all $\mathbf{x} \in \mathbb{R}^n$ (standard basis), we typically represent them as a **n**-component vector.

$$\mathbf{x} : \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_N \end{bmatrix}$$

$$x_i = \frac{\mathbf{x}^T \mathbf{v}_i}{\mathbf{v}_i^T \mathbf{v}_i} = \mathbf{x}^T \mathbf{v}_i$$

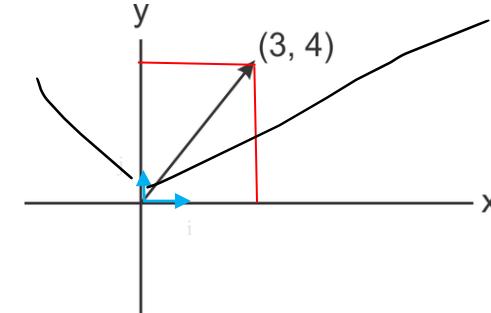
$$\mathbf{x} = \sum_{i=1}^N x_i \mathbf{v}_i = x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 + \dots + x_N \mathbf{v}_N$$



Vector Representation (cont'd)

- Example assuming $n=2$:

$$\mathbf{x} : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$



- Assuming the standard base $\langle v_1=i, v_2=j \rangle$, x_i can be obtained by projecting x along the direction of v_i :

$$x_1 = \mathbf{x}^T i = [3 \quad 4] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 3$$

$$x_2 = \mathbf{x}^T j = [3 \quad 4] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 4$$

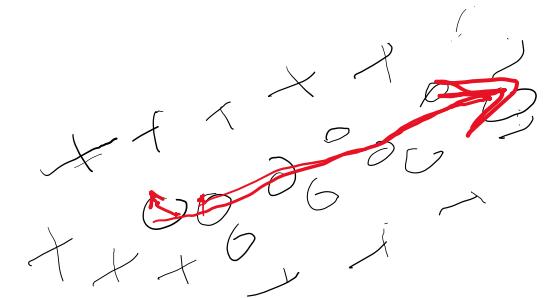
- \mathbf{x} can be “reconstructed” from its projections as follows:

$$\mathbf{x} = 3i + 4j$$

Principal Component Analysis

- The goal in *principal component analysis* (PCA) (also called the *Karhunen-Loeve* or *Hotelling transform*) is to discover the underlying statistical variability in the data and reduce the data's dimensionality from D to a much smaller number of dimensions L ($L \ll D$).
- PCA achieves this goal by
 - Finding the directions of maximum variance in the D -dimensional data
 - Rotating the original coordinate system to align with these directions of maximum variance

Principal Component Analysis



- Most natural signals, including brain signals are redundant
- In the case of EEG measurements from N electrodes
 - Measurements from nearby electrodes may be correlated
 - Underlying rhythms across multiple electrodes.
- PCA attempts to find the dominant directions of variability in the data.
- New data points can be projected along the “principal” directions. Each projection is called a “principal component”
- The resulting L -dimensional vector can be used as a feature vector for classification or other purposes in BCI applications

Principal Component Analysis (PCA)

- If $\mathbf{x} \in \mathbb{R}^N$, then it can be written as a linear combination of an **orthonormal** set of N basis vectors $\langle \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N \rangle$ in \mathbb{R}^N (e.g., using the standard base):

$$\mathbf{v}_i^T \mathbf{v}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{x} = \sum_{i=1}^N x_i \mathbf{v}_i = x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 + \dots + x_N \mathbf{v}_N$$

where $x_i = \frac{\mathbf{x}^T \mathbf{v}_i}{\mathbf{v}_i^T \mathbf{v}_i} = \mathbf{x}^T \mathbf{v}_i$

$$\mathbf{x} : \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$



- PCA seeks to **approximate** \mathbf{x} in a **subspace** of \mathbb{R}^N using a **new** set of $K < N$ basis vectors $\langle \mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K \rangle$ in \mathbb{R}^N :

$$\hat{\mathbf{x}} = \sum_{i=1}^K y_i \mathbf{u}_i = y_1 \mathbf{u}_1 + y_2 \mathbf{u}_2 + \dots + y_K \mathbf{u}_K$$

(reconstruction)

where $y_i = \frac{\mathbf{x}^T \mathbf{u}_i}{\mathbf{u}_i^T \mathbf{u}_i} = \mathbf{x}^T \mathbf{u}_i$

$$\hat{\mathbf{x}} : \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}$$

such that $\|\mathbf{x} - \hat{\mathbf{x}}\|$ is **minimized!**
(i.e., minimize information loss)

Principal Component Analysis (PCA)

- The “**optimal**” set of basis vectors $\langle u_1, u_2, \dots, u_K \rangle$ can be found as follows (we will see why):

(1) Find the **eigenvectors** u_i of the **covariance** matrix of the (training) data Σ_x

$$\Sigma_x u_i = \lambda_i u_i$$

(2) Choose the K “**largest**” eigenvectors u_i (i.e., corresponding to the K “**largest**” eigenvalues λ_i)

$\langle u_1, u_2, \dots, u_K \rangle$ correspond to the “optimal” basis!

We refer to the “**largest**” eigenvectors u_i as **principal components**.

PCA - Steps

- Suppose we are given $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$ ($N \times 1$) vectors

N: # of features

Step 1: compute sample mean

M: # data

$$\bar{\mathbf{x}} = \frac{1}{M} \sum_{i=1}^M \mathbf{x}_i$$

Step 2: subtract sample mean (i.e., center data at zero)

$$\Phi_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

Step 3: compute the sample covariance matrix Σ_x

$$\Sigma_x = \frac{1}{M} \sum_{i=1}^M (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{M} \sum_{i=1}^M \Phi_i \Phi_i^T = \frac{1}{M} A A^T$$

where $A = [\Phi_1 \ \Phi_2 \ \dots \ \Phi_M]$
i.e., the columns of A are the Φ_i
($N \times M$ matrix)

PCA - Steps

Step 4: compute the eigenvalues/eigenvectors of Σ_x

$$\Sigma_x u_i = \lambda_i u_i$$

where we assume $\lambda_1 > \lambda_2 > \dots > \lambda_N$

Note : most software packages return the eigenvalues (and corresponding eigenvectors) in **decreasing** order – if not, you can explicitly put them in this order)

Since Σ_x is symmetric, $\langle u_1, u_2, \dots, u_N \rangle$ form an **orthogonal** basis in R^N and we can represent **any** $x \in R^N$ as:

$$x - \bar{x} = \sum_{i=1}^N y_i u_i = y_1 u_1 + y_2 u_2 + \dots + y_N u_N$$

$$y_i = \frac{(x - \bar{x})^T u_i}{u_i^T u_i} = (x - \bar{x})^T u_i \quad \text{if } \|u_i\| = 1$$

i.e., this is
just a “**change**”
of basis!

$$\begin{matrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{matrix} \rightarrow \begin{matrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{matrix}$$

Note : most software packages **normalize** u_i to unit length to simplify calculations; if not, you can explicitly normalize them)

PCA - Steps

Step 5: dimensionality reduction step – approximate \mathbf{x} using only the **first** K eigenvectors ($K \ll N$) (i.e., corresponding to the **K largest** eigenvalues where K is a **parameter**):

$$\mathbf{x} - \bar{\mathbf{x}} = \sum_{i=1}^N y_i \mathbf{u}_i = y_1 \mathbf{u}_1 + y_2 \mathbf{u}_2 + \dots + y_N \mathbf{u}_N \quad \begin{matrix} \nearrow \\ N \end{matrix} \quad \begin{matrix} \searrow \\ K \end{matrix}$$

approximate \mathbf{x} by $\hat{\mathbf{x}}$
using first K eigenvectors only

$$\hat{\mathbf{x}} - \bar{\mathbf{x}} = \sum_{i=1}^K y_i \mathbf{u}_i = y_1 \mathbf{u}_1 + y_2 \mathbf{u}_2 + \dots + y_K \mathbf{u}_K$$

(reconstruction)

$$\mathbf{x} - \bar{\mathbf{x}}: \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_N \end{bmatrix} \rightarrow \hat{\mathbf{x}} - \bar{\mathbf{x}}: \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_K \end{bmatrix}$$

note that if $\underbrace{K=N}$, then $\hat{\mathbf{x}} = \mathbf{x}$
(i.e., zero reconstruction error)

What is the Linear Transformation implied by PCA?

- The linear transformation $y = Tx$ which performs the dimensionality reduction in PCA is:

$$\hat{\mathbf{x}} - \bar{\mathbf{x}} = \sum_{i=1}^K y_i u_i = y_1 u_1 + y_2 u_2 + \dots + y_K u_K$$

$$(\hat{\mathbf{x}} - \bar{\mathbf{x}}) = U \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_K \end{bmatrix}$$

where $U = [u_1 \ u_2 \ \dots \ u_K]$ $N \times K$ matrix
i.e., the columns of U are the the first K eigenvectors of Σ_x


$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_K \end{bmatrix} = U^T (\hat{\mathbf{x}} - \bar{\mathbf{x}})$$

$T = U^T$ $K \times N$ matrix
i.e., the rows of T are the first K eigenvectors of Σ_x

What is the form of Σ_y ?

$$\Sigma_x = \frac{1}{M} \sum_{i=1}^M (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{M} \sum_{i=1}^M \Phi_i \Phi_i^T$$

Using diagonalization:

$$\Sigma_x = P \Lambda P^T$$

The columns of P are the
eigenvectors of Σ_x

The diagonal elements of
 Λ are the **eigenvalues** of Σ_x
or the **variances**

$$\mathbf{y}_i = U^T (\mathbf{x}_i - \bar{\mathbf{x}}) = P^T \Phi_i$$

$$\Sigma_y = \frac{1}{M} \sum_{i=1}^M (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T = \frac{1}{M} \sum_{i=1}^M (\mathbf{y}_i)(\mathbf{y}_i)^T = \frac{1}{M} \sum_{i=1}^M (P^T \Phi_i)(P^T \Phi_i)^T =$$

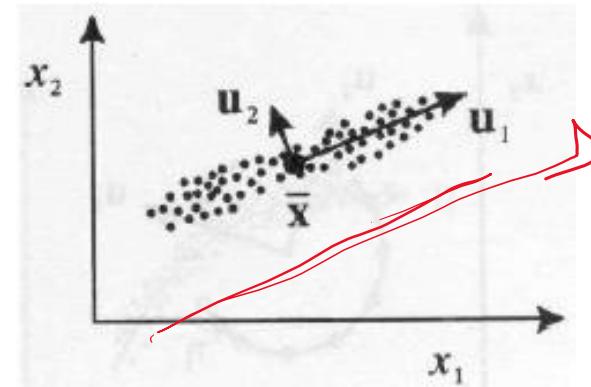
$$\frac{1}{M} \sum_{i=1}^M (P^T \Phi_i)(\Phi_i^T P) = P^T \left(\frac{1}{M} \sum_{i=1}^M \Phi_i \Phi_i^T \right) P = P^T \Sigma_x P = P^T (P \Lambda P^T) P = \Lambda$$

$$\Sigma_y = \Lambda$$

PCA de-correlates the data!
Preserves original variances!

Interpretation of PCA

- PCA chooses the **eigenvectors** of the covariance matrix corresponding to the **largest** eigenvalues.
- The **eigenvalues** correspond to the **variance** of the data along the eigenvector directions.
- Therefore, PCA projects the data along the directions where the data varies **most**.
- PCA preserves as much **information** in the data by preserving as much **variance** in the data.



u_1 : direction of **max** variance
 u_2 : orthogonal to u_1

Example

- Compute the PCA of the following dataset:

(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)

- Compute the sample covariance matrix is:

$$\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^n (\mathbf{x}_k - \hat{\mu})(\mathbf{x}_k - \hat{\mu})^t$$

$$\Sigma_x = \begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}$$

- The eigenvalues can be computed by finding the roots of the characteristic polynomial:

$$\begin{aligned}\Sigma_x v &= \lambda v \Rightarrow |\Sigma_x - \lambda I| = 0 \\ &\Rightarrow \begin{vmatrix} 6.25 - \lambda & 4.25 \\ 4.25 & 3.5 - \lambda \end{vmatrix} = 0 \\ &\Rightarrow \lambda_1 = 9.34; \lambda_2 = 0.41\end{aligned}$$

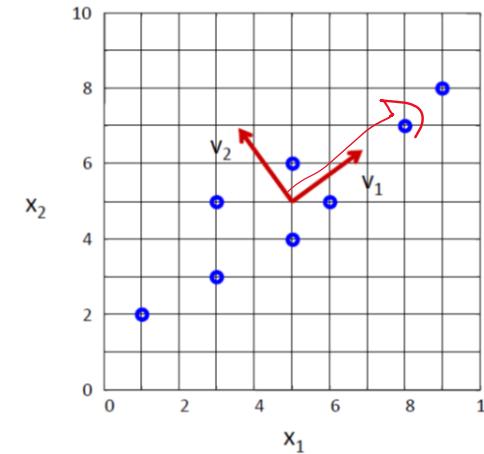
Example (cont'd)

- The eigenvectors are the solutions of the systems:

$$\sum_{\mathbf{x}} \mathbf{u}_i = \lambda_i \mathbf{u}_i$$



$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} \lambda_1 v_{11} \\ \lambda_1 v_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix}$$
$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} \lambda_2 v_{21} \\ \lambda_2 v_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix}$$



Note: if \mathbf{u}_i is a solution, then $c\mathbf{u}_i$ is also a solution where $c \neq 0$.

Eigenvectors can be normalized to unit-length using:

$$\hat{\mathbf{v}}_i = \frac{\mathbf{v}_i}{\| \mathbf{v}_i \|}$$

How do we choose K ?

- K is typically chosen based on how much **information (variance)** we want to preserve:

Choose the **smallest**
 K that satisfies
the following
inequality:

$$\frac{\sum_{i=1}^K \lambda_i}{\sum_{i=1}^N \lambda_i} > T \quad \text{where } T \text{ is a threshold (e.g., 0.9)}$$

- If $T=0.9$, for example, we “**preserve**” 90% of the information (variance) in the data.
- If $K=N$, then we “**preserve**” 100% of the information in the data (i.e., just a “**change**” of basis and $\hat{\mathbf{x}} = \mathbf{x}$)

Approximation Error

- The approximation error (or reconstruction error) can be computed by:

$$\| \mathbf{x} - \hat{\mathbf{x}} \|$$

where $\hat{\mathbf{x}} = \sum_{i=1}^K y_i u_i + \bar{\mathbf{x}} = y_1 u_1 + y_2 u_2 + \dots + y_K u_K + \bar{\mathbf{x}}$
(reconstruction)

- It can also be shown that the approximation error can be computed as follows:

$$\| \mathbf{x} - \hat{\mathbf{x}} \| = \frac{1}{2} \sum_{i=K+1}^N \lambda_i$$

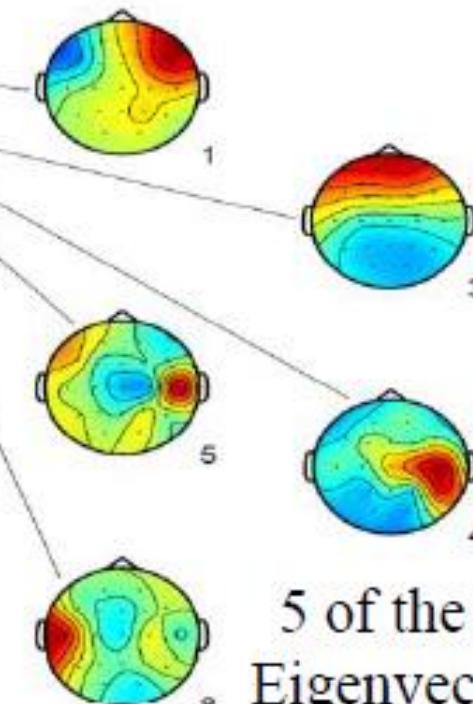
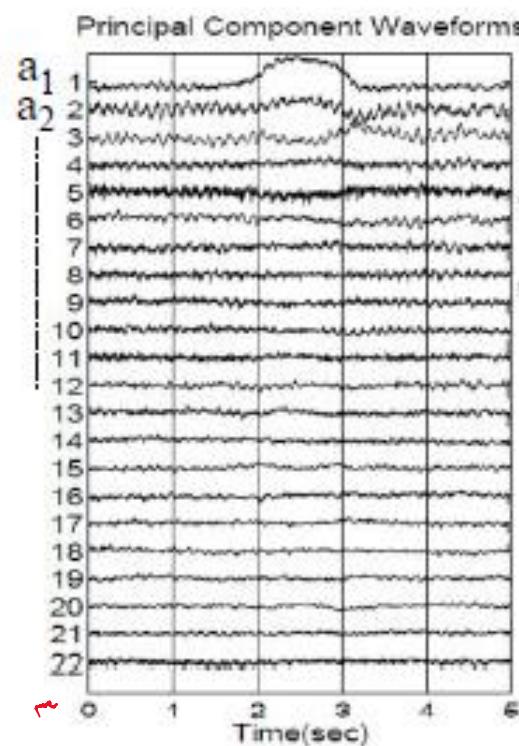
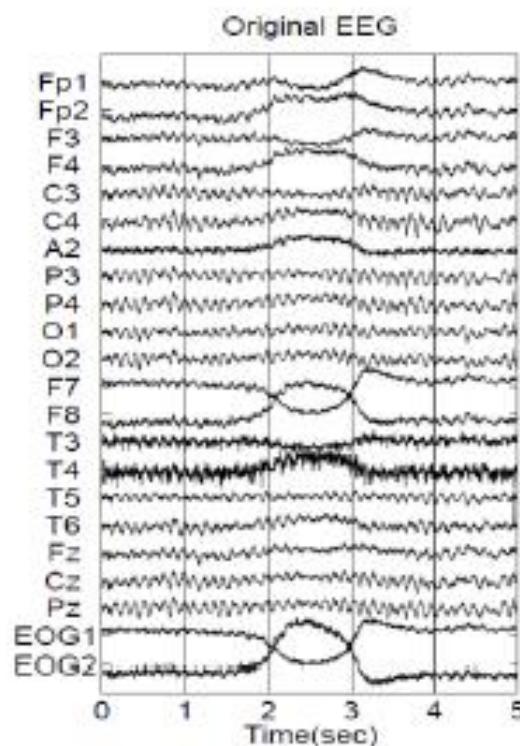


Data Normalization

- The principal components are dependent on the *units* used to measure the original variables as well as on the *range* of values they assume.
- Data should **always** be normalized prior to using PCA.
- A common normalization method is to transform all the data to have **zero mean** and **unit standard deviation**:

$$\frac{x_i - \mu}{\sigma} \quad \text{where } \mu \text{ and } \sigma \text{ are the mean and standard deviation of the } i\text{-th feature } x_i$$

PCA applied to EEG



5 of the 22
Eigenvectors
(spatial filters
for EEG
channels)

(Jung et al., 1998)

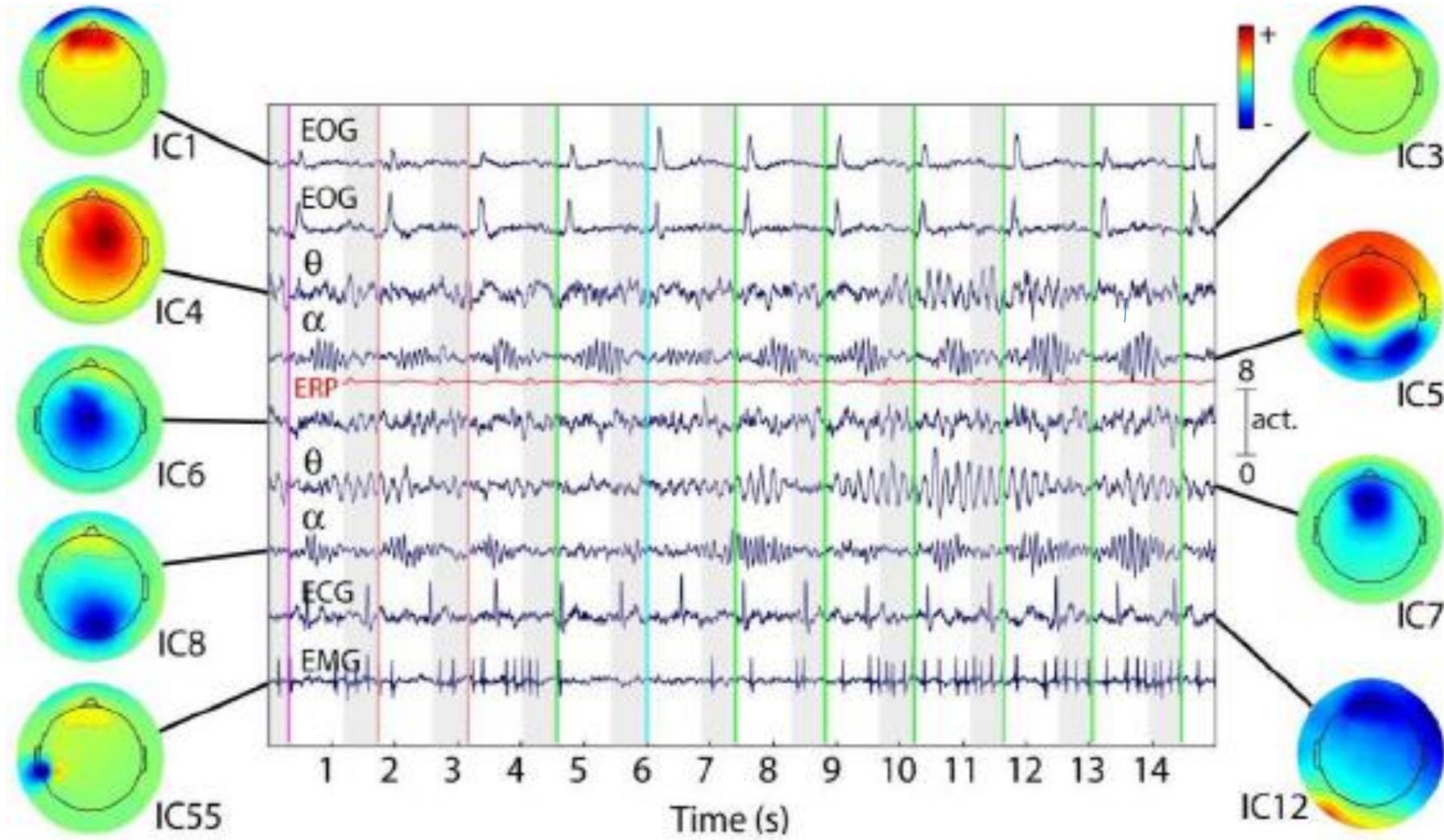
Independent Component Analysis

- PCA finds a matrix \mathbf{V} that decorrelates the inputs but the resulting feature vector \mathbf{a} may still retain higher order statistical dependencies
- There may be a possibility that the variables are independent.
- ICA tries to find a matrix \mathbf{W} of filters (columns of \mathbf{W}) such that the output \mathbf{a} is **statistically independent**:

$$\mathbf{a} = \mathbf{W}^T \mathbf{x} \text{ such that } P(\mathbf{a}) \approx \prod_{i=1}^D P(a_i)$$

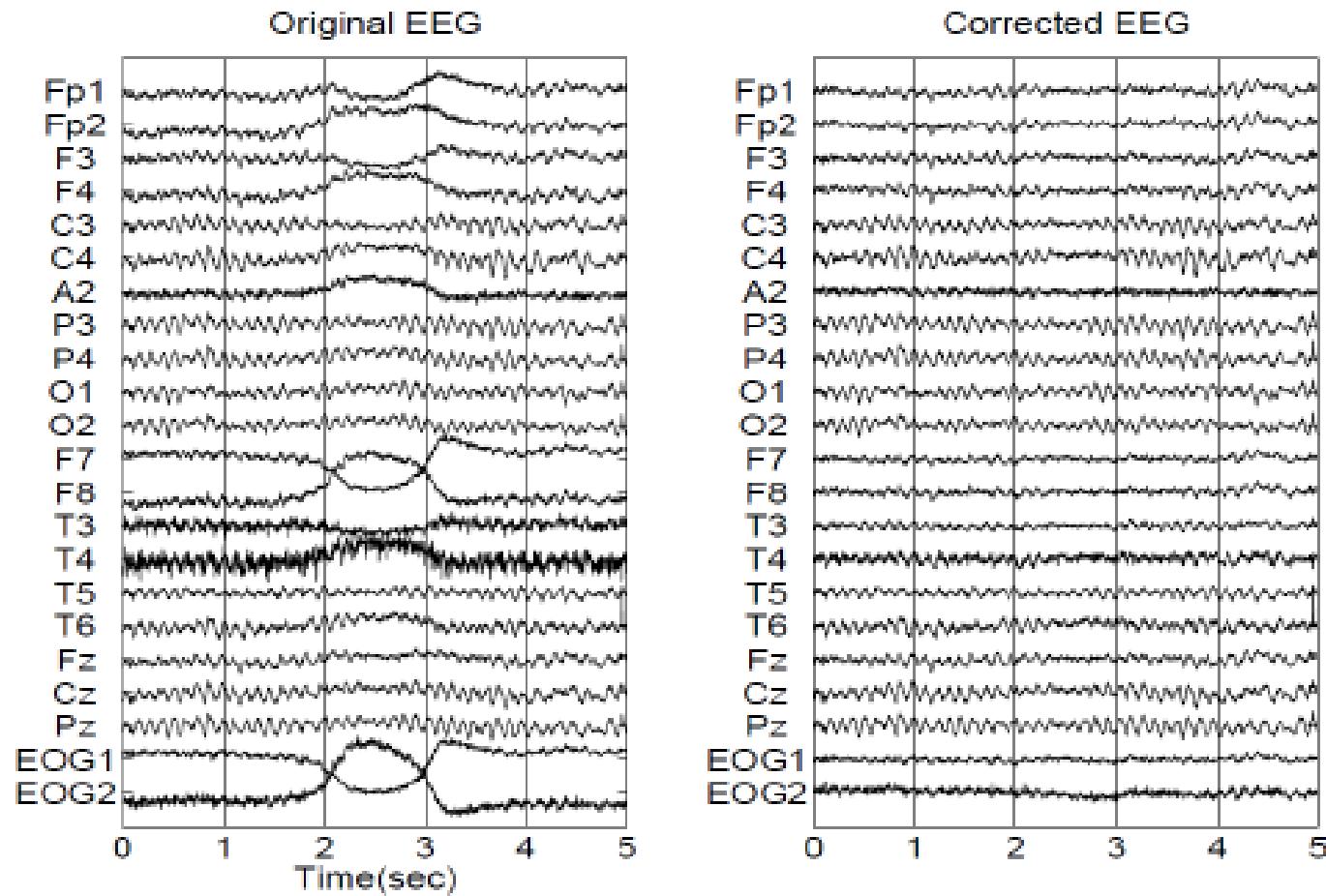
Independent Component Analysis

- ICA assumes sources are linearly mixed to produce x
- The feature vector dimension in ICA can be lesser than, equal to, or greater than the number of input dimensions.
- ICA has proved useful in a variety of settings in BCI applications, ranging from the use of the output vector **a** as a feature vector in classification.



Application of ICA to EEG data for isolating electro- oculographic (EOG) (eye-related), electromyographic (EMG) (muscle-related) and electrocardiographic (ECG) (heart-related) artifacts, and unmixing putative source signals in the brain. Image (adapted from Onton and Makeig, 2006)

ICA for Artifact Removal in EEG



Common Spatial Pattern

- Supervised Technique
- Data is labeled with class to which each data vector belongs
 - E.g., EEG obtained for right versus left hand imagery
- CSP finds a matrix of spatial filters
 - the variance of the filtered data for one class is maximized
 - variance of the filtered data for the other class is minimized
- CSP filters can significantly enhance discrimination ability between the two classes

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \\ \vdots & \vdots & \vdots \end{bmatrix}_{N \times T}$$

Common Spatial Pattern

Input: $\{X_c^i\}_{i=1}^K$ C : classes, i : trial, K : no. of trials.

$X_c^i \rightarrow N \times T$ [No. of channels + Time samples]

Assuming X_c^i is centered & scaled

Goal: - Spatial filter Matrix ' W ' $\rightarrow N \times M$

Transform signal according to eq :-

$$x_{CSPL}(t) = W^T x(t) \quad \text{--- (1)}$$

X -Matrix

DC-Vector

Common Spatial Pattern

Two class conditional covariance Matrix

$$R_c = \frac{1}{K} \sum_i x_c^i (x_c^i)^T \quad \text{for } c \in [1, 2]$$

determining w as

$$w^T R_1 w = \lambda_1$$

$$w^T R_2 w = \lambda_2$$

$$\begin{bmatrix} \lambda_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \lambda_2 \end{bmatrix}$$

$$\lambda_1 + \lambda_2 = I$$

$$R_1 \omega = \lambda R_2 \omega$$

Generalized eigenvalue

$$\lambda_1^j = \omega_j^T R_1 \omega_j \rightarrow \gamma_1$$

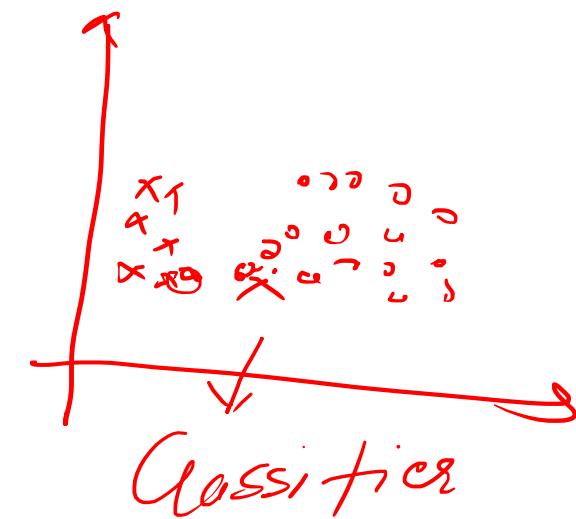
$$\lambda_2^j = \omega_j^T R_2 \omega_j \rightarrow \gamma_2$$

$$\lambda_1^j + \lambda_2^j = 1$$

High value
High variance

Low value

Low variance



CSP applied to EEG for Right/Left Hand Imagery

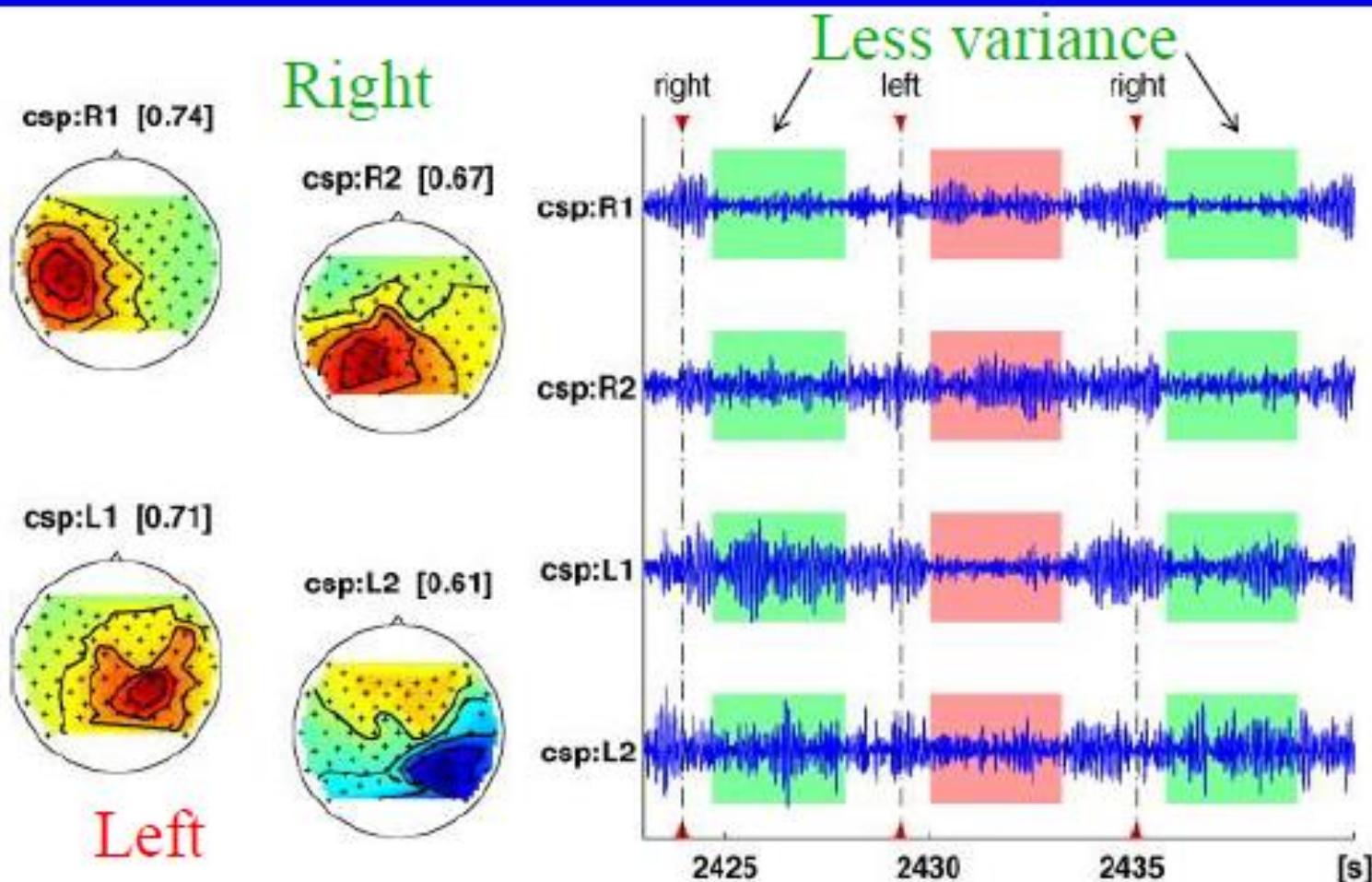


Image source: <http://www.sciencedirect.com/science/article/pii/S0165027007004657>

Artifact Reduction Techniques

- Artifacts in BCIs are any undesirable signals
 - Artifacts outside body-50/60Hz
 - Power line
 - External electrical interference
 - Artifacts within body
 - Rhythmic artifacts due to respiration and heartbeat (the latter are called electrocardiographic or ECG artifacts)
 - Signal distortion or attenuation due to skin conductance changes
 - Eye movement and eye blink artifacts (also called electro-oculographic or EOG artifacts)-- range 3–4Hz
 - Muscle artifacts (electromyographic or EMG artifacts)-- 30Hz or higher frequency range.

Artifact Reduction Techniques

- Thresholding
 - If the magnitude or some other characteristic of a recorded EOG or EMG signal exceeds a pre-determined threshold, the brain signals recorded during that epoch are deemed to be contaminated and rejected.
- Band-Stop and Notch Filtering
 - Band-stop filtering is a useful artifact reduction technique that attenuates the components of a signal in a specific frequency band and passes the rest of the components of the signal.
 - A notch filter set to the 59–61 Hz band (in the United States) for filtering out the 60 Hz power-line noise artifact.

Artifact Reduction Techniques

- Linear Modeling
 - A simple way of modeling the effect of artifacts on a recorded brain signal is to assume that the effect is additive.
 - For example, if $EEG_i(t)$ is the EEG signal recorded from electrode i at time t , then a model of how the signal has been contaminated could be:

$$EEG_i(t) = EEG_i^{true}(t) + K \cdot EOG(t)$$

- $EEG_i^{true}(t)$ is the uncontaminated (“true”) EEG signal from electrode i at time t , $EOG(t)$ is the recorded EOG signal at time t and K is a constant.
- Given an estimated value for K , one can obtain an estimate of the true EEG signal using:

$$EEG_i^{true}(t) = EEG_i(t) - K \cdot EOG(t)$$



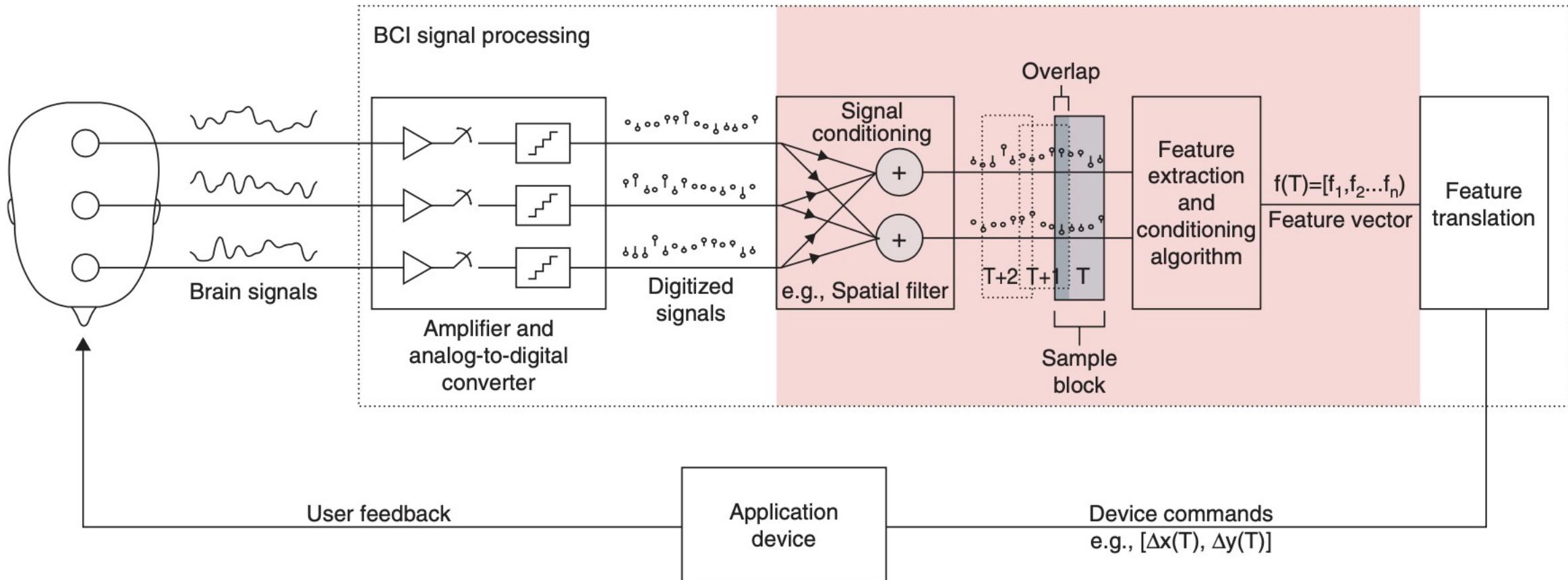
Brain Computer Interaction

Feature Extraction

Features

- The purpose of a BCI is to **detect and quantify characteristics of brain signals** that indicate what the user wants the BCI to do, to translate these measurements in real time into the desired device commands, and to provide concurrent feedback to the user.
- The brain-signal characteristics used for this purpose are called ***signal features, or simply features.***
- ***Feature extraction*** is the process of distinguishing the pertinent signal characteristics from extraneous content and representing them in a compact and/or meaningful form, amenable to interpretation by a human or computer.

Overall structure of a BCI



Feature vector

- A **fundamental signal feature** is simply a direct measurement of the signal. They usually provide limited relevant information about typically complex brain signals.
- Thus, it is more common for BCIs to use features that are **linear or nonlinear combinations, ratios, statistical measures, or other transformations of multiple fundamental features** detected at multiple electrodes and/or multiple time points.
- Such complex features, if selected appropriately, can reflect the user's desires more accurately than the fundamental features themselves.
- Most features used in BCI applications are based on **spatial, temporal, and/or spectral analyses** of brain signals or the relationships among them.
- Furthermore, in order to determine the user's wishes as accurately as possible, most BCIs extract a number of features simultaneously. This set of features is referred to as a ***feature vector***.

Feature vector

To be effective for BCI applications, a feature should have the following attributes:

- its spatial, temporal, spectral characteristics, and dynamics can be precisely characterized for an individual user or population of users
- it can be modulated by the user and used in combination with other features to reliably convey the user's intent
- its correlation with the user's intent is stable over time and/or can be tracked in a consistent and reliable manner

BCI SIGNAL PROCESSING- Fourier Analysis

- Much of signal-processing theory is rooted in Fourier analysis, which transforms a time-domain (i.e., time on the x-axis) signal into its equivalent frequency-domain (i.e., frequency on the -axis) representation.
- The primary utility of Fourier analysis is to decompose a signal into individual sinusoidal components that can be isolated and evaluated independently.
- Using Fourier analysis, practically any signal can be accurately represented as the sum of a number (possibly an infinite number) of amplitude-scaled and time-shifted sinusoids at specific frequencies.
- In order to model these signals, it is necessary to properly adjust the phase and the magnitude of each sinusoid.

Fourier Analysis

- For an arbitrary signal $x(t)$, the magnitude (scale) and phase (shift) of the sinusoid at each frequency [ω (radians) = $2\pi f$ (Hz)] required to represent an arbitrary signal can be determined from the Fourier transform:

$$\begin{aligned} X(\omega) &= \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt = \int_{-\infty}^{\infty} x(t)[\cos\omega t + j\sin\omega t]dt \\ &= \underbrace{\int_{-\infty}^{\infty} x(t)\cos\omega t dt}_{a(\omega)} + j \underbrace{\int_{-\infty}^{\infty} x(t)\sin\omega t dt}_{b(\omega)} \\ &= a(\omega) + jb(\omega) \end{aligned}$$

The magnitude and phase for each sinusoidal component are given as:

$$Magnitude : |X(\omega)| = \sqrt{a^2(\omega) + b^2(\omega)}$$

$$Phase : \theta = \arg(X(\omega)) = \tan^{-1}\left(\frac{b(\omega)}{a(\omega)}\right)$$

BCI SIGNAL PROCESSING – Digital Filtering

- Digital filters are central to digital signal processing. They modify the frequency content of a digital signal by attenuating some frequencies (or frequency ranges) and amplifying others. Each successive sample of a digitized signal is passed through a digital filter to produce a new value as the output.
 - Low pass filter (higher frequencies are attenuated and lower frequencies are preserved)
 - High pass filter (very specific ranges of signal frequencies can be amplified, attenuated, preserved, and/or eliminated)
 - Bandpass filter (A band-pass filter preserves signal power within a specified continuous frequency range, while attenuating signal power outside of this range)
 - Notch filter (A notch filter is the converse of a bandpass filter; it attenuates signal power within a specified continuous frequency range, while preserving signal power outside of this range)

THE THREE STEPS OF FEATURE EXTRACTION

The process of feature extraction is discussed here as a three-step procedure:

- signal conditioning to reduce noise and to enhance relevant aspects of the signals
- extraction of the features from the conditioned signals
- feature conditioning to properly prepare the feature vector for the feature-translation stage

FIRST STEP: SIGNAL CONDITIONING

- The first step of feature extraction is called signal conditioning or preprocessing.
- This step enhances the signal by preemptively eliminating known interference (i.e., artifacts) or irrelevant information, and/or by enhancing spatial, spectral, or temporal characteristics of the signal that are particularly relevant to the application.
- It is common to have some prior knowledge about the general signal characteristics relevant for a particular application, and this knowledge is used in conditioning.

FIRST STEP: SIGNAL CONDITIONING

Signal conditioning can include a number of different procedures that can primarily be categorized as:

- frequency-range prefiltering
- data decimation and normalization
- spatial filtering
 - Data independent spatial filtering (*common- average reference and surface Laplacian spatial filters*)
 - Data dependent spatial filtering (PCA, ICA and CSP)
- removal of environmental interference and biological artifacts

SECOND STEP: EXTRACTING THE FEATURES

- ***BLOCK PROCESSING***

- For most BCI applications, it is highly desirable for the processing to occur in real time. Prior to feature extraction, the incoming signal samples are commonly segmented into consecutive, possibly overlapping, sample blocks.
- A feature vector is created from the signal samples within each individual sample block. The feature vectors from the successive sample blocks are then fed to the translation algorithm, which produces a device command or user feedback corresponding to each sample block or corresponding to sets of consecutive sample blocks.
- For efficient online implementation, the length and overlap of these sample blocks should fit the relevant temporal dynamics of the signal, the feature-extraction method, the nature of the application, and the concurrent user feedback, as well as the available processing power.
 - E.g., BCI cursor control
 - P300 response

SECOND STEP: EXTRACTING THE FEATURES

- ***TIME (TEMPORAL) FEATURES***

1. Peak-Picking and Integration

- Peak-picking simply determines the minimum or maximum value of the signal samples in a specific time block (usually defined relative to a specific preceding stimulus) and uses that value (and possibly its time of occurrence) as the feature(s) for that time block.
- The signal can be averaged or integrated over all or part of the time block to yield the feature(s) for the block. Some form of averaging or integration is typically preferable to simple peak-picking, especially when the responses to the stimulus are known to vary in latency and/or when unrelated higher-frequency activity is superimposed on the relevant feature

SECOND STEP: EXTRACTING THE FEATURES

- ***TIME (TEMPORAL) FEATURES***
2. Correlation and Template-Matching
 - The similarity of a response to a predefined template might also be used as a feature.

SECOND STEP: EXTRACTING THE FEATURES

Statistical features

Sl. No	Features	Short description
1	MEAN	Mean value
2	STD	Standard deviation
3	MAX VALUE	Maximum positive amplitudes
4	MIN VALUE	Maximum negative amplitudes
5	SKEWNESS	a measure of asymmetry of the distribution
6	KURTOSIS	a measure of flatness of the distribution
7	MEDIAN	the middle value of a set of ordered data

SECOND STEP: EXTRACTING THE FEATURES

Interval or period analysis features.

Sl. No	Features	Short description
8	LINE LENGTH	Line length
9	MEAN VV AMPL	Mean of vertex-to-vertex amplitudes
10	VAR VV AMPL	Variance of vertex-to-vertex amplitudes
11	MEAN VV TIME	Mean of vertex-to-vertex times
12	VAR VV TIME	Variance of vertex-to-vertex times
13	MEAN VV SLOPE	Mean of vertex-to-vertex slope
14	VAR VV SLOPE	Variance of vertex-to-vertex slope
15	ZERO CROSSING	Number of zero crossings in a signal
16	MIN MAX NUMBER	Number of local minima and maxima
17	COEFF OF VARIATION	a statistical measure of the deviation of a variable from its mean, standard deviation divided by mean
18	AMPL RANGE	The difference between the maximum positive and maximum negative Amplitude values

SECOND STEP: EXTRACTING THE FEATURES

Features derived from the first and second derivative.

Sl. No	Features	Short description
19	1 st DIFF MEAN	Mean value of the first derivative of the signal
20	1 st DIFF MAX	Maximum value of the first derivative of the signal
21	2 nd DIFF MEAN	Mean value of the second derivative of the signal
22	2 nd DIFF MAX	Maximum value of the second derivative of the signal

SECOND STEP: EXTRACTING THE FEATURES

The Hjorth parameters

Sl. No	Features	Short description
23	HJORTH 1	Ability
24	HJORTH 2	Mobility ($\sigma x' / \sigma x$)
25	HJORTH 3	Complexity

NOTE:

$\sigma x'$ is the standard deviation of the first derivative of the signal
 $\sigma x''$ is the standard deviation of the second derivative of the signal

SECOND STEP: EXTRACTING THE FEATURES

- ***FREQUENCY (SPECTRAL) FEATURES***
- Much brain activity manifests itself as continuous amplitude- and frequency-modulated oscillations. Therefore, it is often advantageous to accurately track these changes in the frequency domain. Although the Fourier transform is the most common method for converting from the time domain to the frequency domain, there are several alternatives that have characteristics that are particularly desirable given specific constraints or specific objectives. These include:
 - band power
 - fast Fourier transform (FFT)
 - autoregressive (AR) modeling

SECOND STEP: EXTRACTING THE FEATURES

FFT-based features calculated from the EEG spectra.

Sl. No	Features	Short description
26	FFT DELTA	0.1 - 3 Hz
27	FFT THETA	3 - 7 Hz
28	FFT ALPHA	7 - 12 Hz
29	FFT BETA	12 - 30 Hz
30	FFT GAMMA	30 - 40 Hz
31	FFT WHOLE	0.1 - 40 Hz

SECOND STEP: EXTRACTING THE FEATURES

FFT-based Spectral Features.

Sl. No	Features	Short description
32	FFT DT RATIO	<i>DELTA / THETA</i>
33	FFT DA RATIO	<i>DELTA / ALPHA</i>
34	FFT TA RATIO	<i>THETA / ALPHA</i>
35	FFT DTA RATIO	<i>(DELTA + THETA) / ALPHA</i>
36	FFT SEF	Spectral edge frequency (95 % of the total spectral power resides)
37	FFT SP-ROLL OFF	The frequency below which 85 % of the total spectral power resides

SECOND STEP: EXTRACTING THE FEATURES

Wavelet based Features.

Sl. No	Features	Short description
38	MIN WAV VALUE	Minimum value
39	MAX WAV VALUE	Maximum value
40	MEAN WAV VALUE	Mean value
41	MEDIAN WAV VALUE	Median value
42	STD WAV VALUE	Standard deviation
43	SKEWNESS WAV VALUE	Skewness
44	KURTOSIS WAV VALUE	Kurtosis
45	WAV BAND	Relative energy

SECOND STEP: EXTRACTING THE FEATURES

Wavelet based Features.

Sl. No	Features	Short description
46	ENTROPY SPECTRAL WAV	The spectral entropy
47	1 st DIFF WAV MEAN	Mean value of the 1 st derivative
48	1 st DIFF WAV MAX	Maximum value of the 1 st derivative
49	2 nd DIFF WAV MEAN	Mean value of the 2 nd derivative
50	2 nd DIFF WAV MAX	Maximum value of the 2 nd derivative
51	ENERGY PERCENT WAV	Percentage of the total energy of a detail/approximation
52	WAV ZERO CROSSING	Zero crossing
53	WAV COEFF OF VARIATION	Coefficient of variation
54	WAV TOTAL ENERGY	Total Energy

SECOND STEP: EXTRACTING THE FEATURES

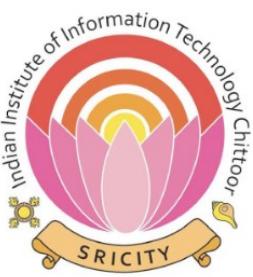
Other Features.

Sl. No	Features	Short description
55	ENTROPY SPECTRAL	The spectral entropy
56	ENTROPY SHANNON	The Shannon entropy
57	MAX ABS XCORR EEG-EEG	Maximum positive amplitude of auto-correlation or cross-Correlation function
58	MEAN ABS XCORR EEG-EEG	Mean value of auto-correlation or cross-correlation function

THIRD STEP: FEATURE CONDITIONING

- The distributions and the relationships among the features can have a significant effect on the performance of the translation algorithm that follows feature extraction. These effects depend on the characteristics of the particular translation algorithm.
 - ***NORMALIZATION***
 - ***LOG-NORMAL TRANSFORMS***
 - ***FEATURE SMOOTHING***
 - ***PCA AND ICA***

Thank You!



Brain Computer Interaction

Feature Translation

Course Instructors

Dr. Annushree Bablani

Acknowledgements: Dr. Sreeja S R

THIRD STEP: FEATURE CONDITIONING

- The distributions and the relationships among the features can have a significant effect on the performance of the translation algorithm that follows feature extraction. These effects depend on the characteristics of the particular translation algorithm.
- ***NORMALIZATION***
- ***LOG-NORMAL TRANSFORMS***
- ***FEATURE SMOOTHING***
- ***PCA AND ICA***
- ***Removing irrelevant and redundant features (MRMR-Maximum Relevant Minimum Redundant)***

Feature Translation

- Discriminant functions
- Regression functions

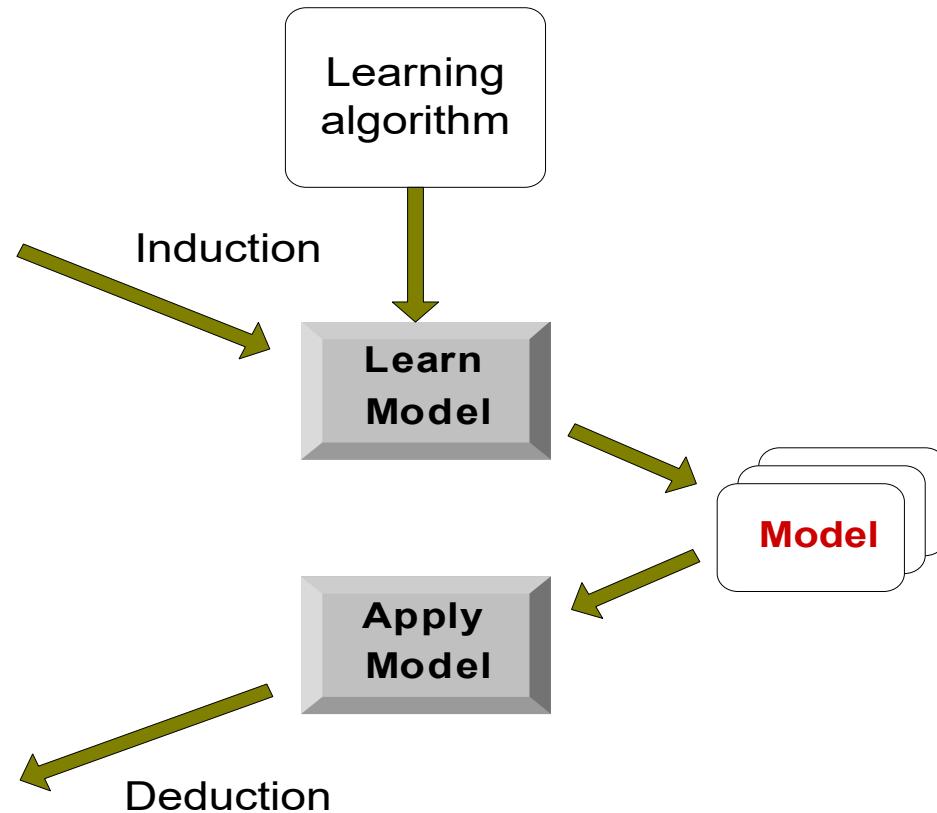
Illustrating Classification Tasks

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

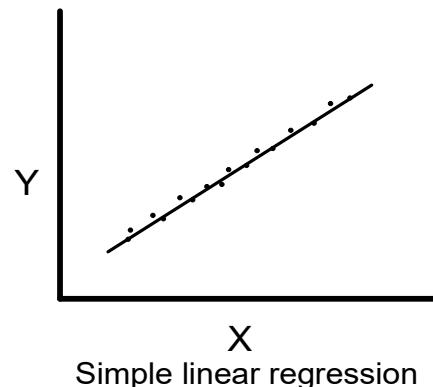
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

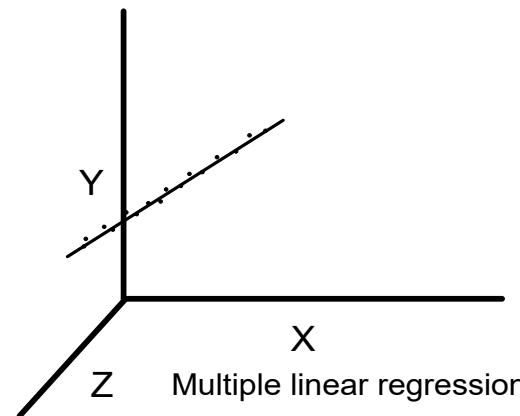


Regression Analysis

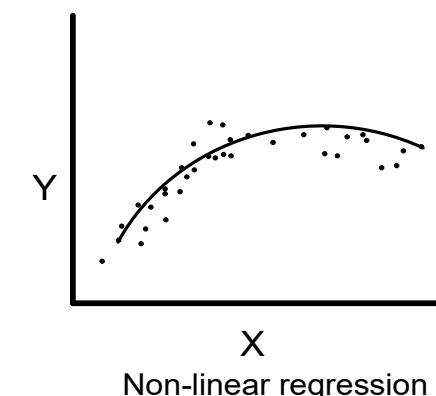
- The regression analysis is a statistical method to deal with the formulation of mathematical model depicting relationship amongst variables, which can be used for the purpose of prediction of the values of dependent variable, given the values of independent variables.
- Classification of Regression Analysis Models**
 - Linear regression models
 - Simple linear regression
 - Multiple linear regression
 - Non-linear regression models



Simple linear regression



Multiple linear regression

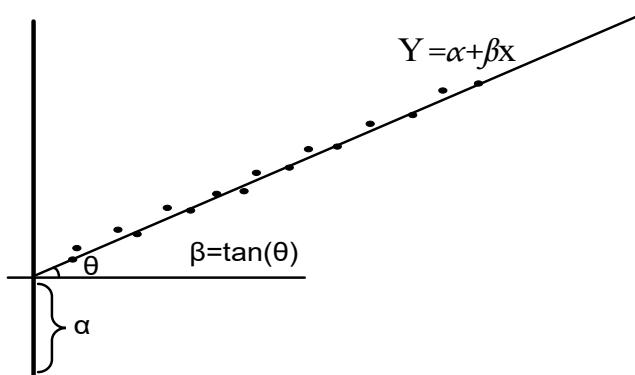


Non-linear regression

Simple Linear Regression Model

In simple linear regression, we have only two variables:

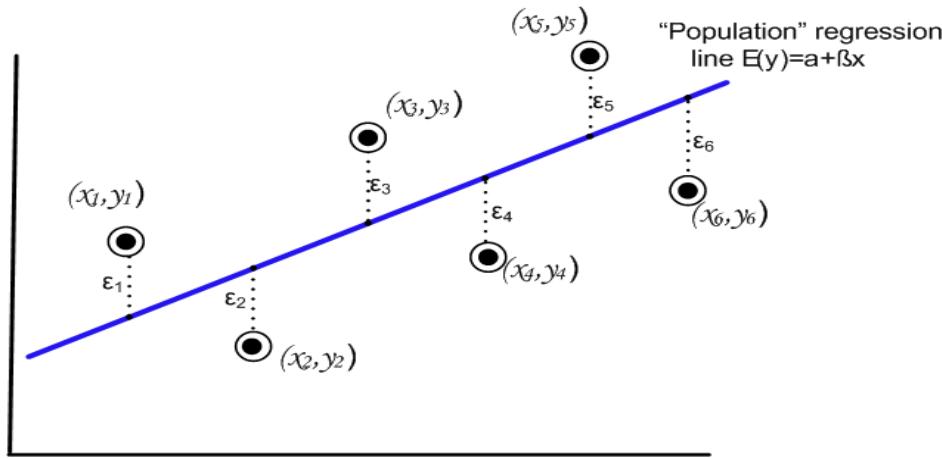
- Dependent variable (also called **Response**), usually denoted as .
- Independent variable (alternatively called **Regressor**), usually denoted as .
- A reasonable form of a relationship between the Response and the Regressor is the linear relationship, that is in the form



Note:

- There are infinite number of lines (and hence)
- The concept of regression analysis deal with finding the best relationship between and (and hence best fitted values of) quantifying the strength of that relationship.

Regression Analysis



Given the set of data involving pairs of values, our objective is to find “true” or population regression line such that

Here, is a random variable with and . The quantity is often called the **error variance**.

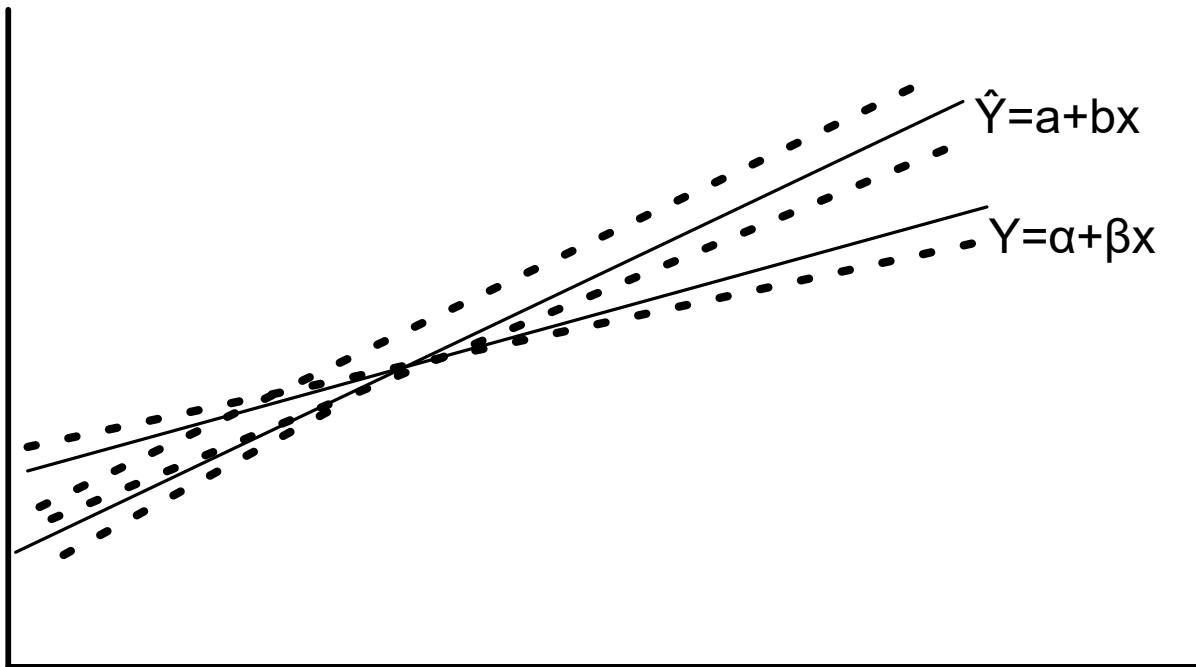
Note:

- implies that at a specific , the values are distributed around the “true” regression line (i.e., the positive and negative errors around the true line is reasonable).
- are called **regression coefficients**.
- values are to be estimated from the data.

True versus Fitted Regression Line

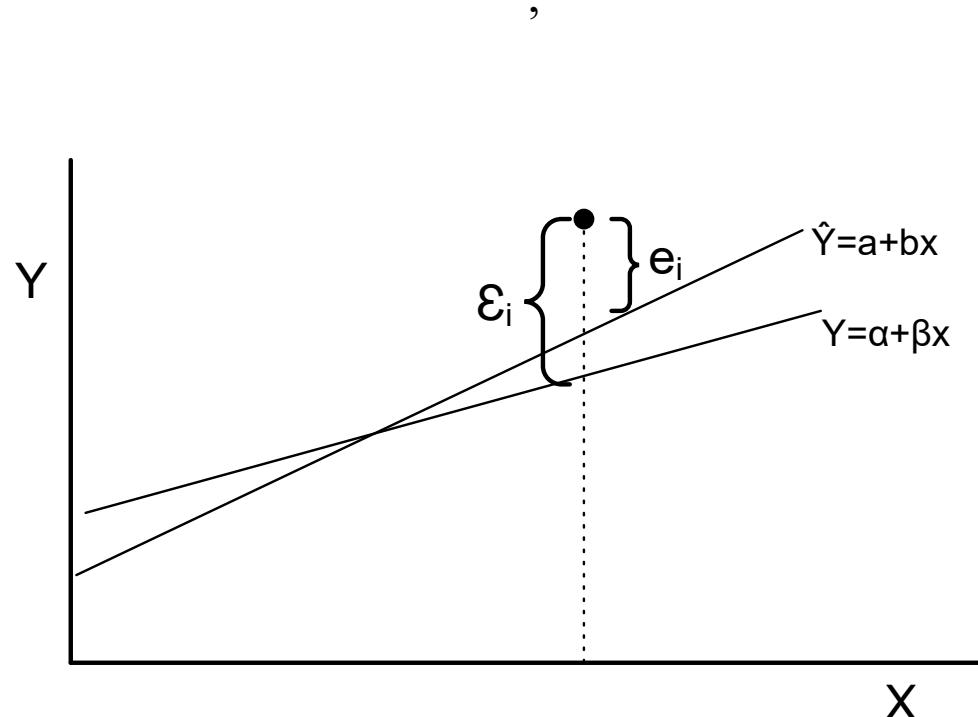
- The task in regression analysis is to estimate the regression coefficients .
- Suppose, we denote the estimates a for α and b for β . Then the fitted regression line is

where \hat{Y} is the predicted or fitted value.



Least Square Method to estimate

This method uses the concept of **residual**. A residual is essentially an error in the fit of the model . Thus, residual is



Least Square method

- The residual sum of squares is often called **the sum of squares of the errors** about the fitted line and is denoted as SSE

$$\text{SSE} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- We are to minimize the value of SSE and hence to determine the parameters of a and b .
- Differentiating SSE with respect to a and b , we have

For minimum value of SSE, $\frac{\partial \text{SSE}}{\partial a} = 0$

$$\frac{\partial \text{SSE}}{\partial b} = 0$$

Least Square method to estimate

Thus we set

$$+b =$$

These two equations can be solved to determine the values of a and b , and it can be calculated that



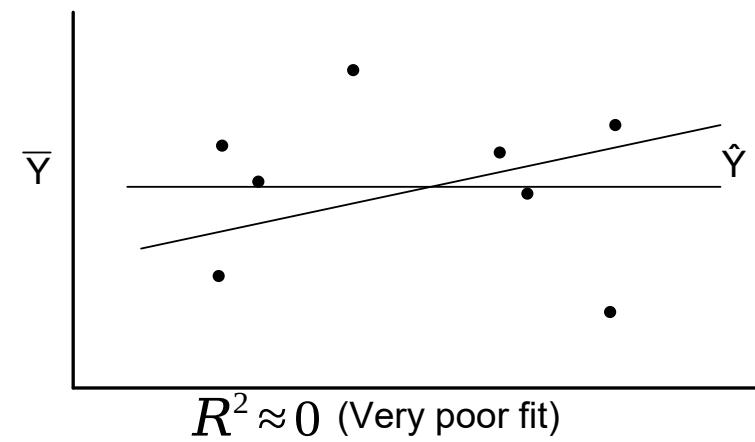
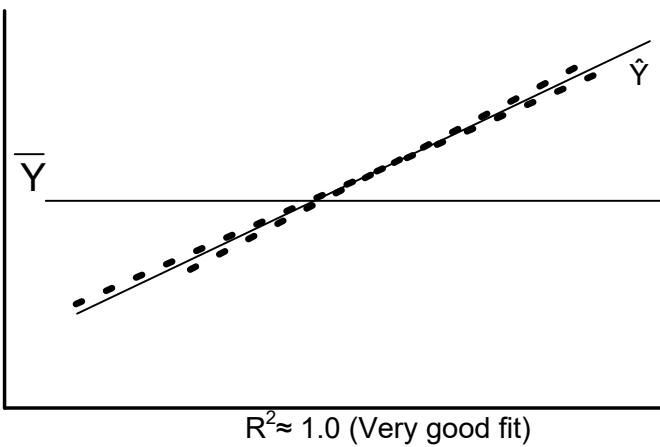
: Measure of Quality of Fit

- A quantity , is called **coefficient of determination** is used to measure the proportion of variability of the fitted model.
- We have
- It signifies the **variability due to error**.
- Now, let us define the **total corrected sum of squares**, defined as
- SST represents the variation in the response values. The is

Note:

- If fit is perfect, all residuals are zero and thus $R^2 = 1.0$ (very good fit)
- If SSE is only slightly smaller than SST, then $R^2 \approx 0$ (very poor fit)

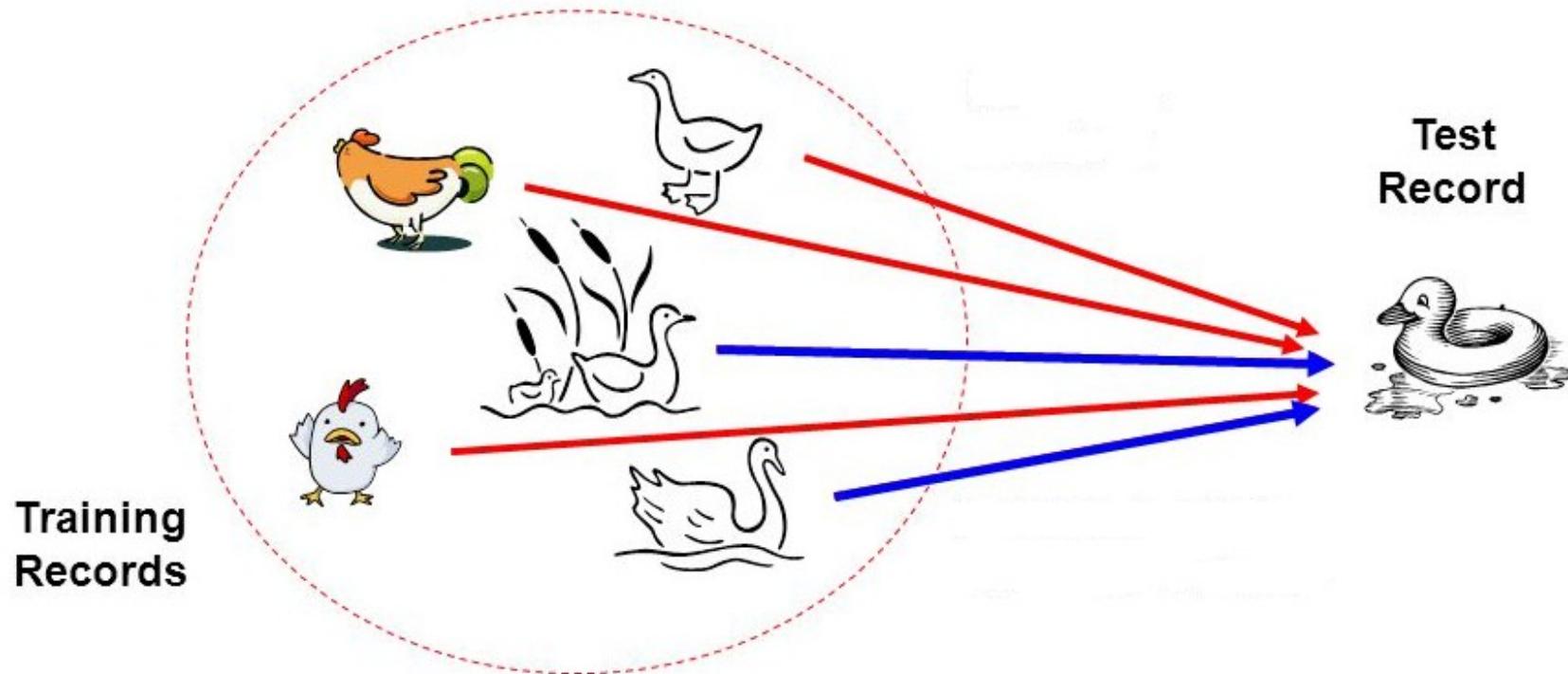
: Measure of Quality of Fit



Bayesian Classifier

Bayesian Classifier

- Principle
 - If it walks like a duck, quacks like a duck, then it is **probably** a duck



Bayesian Classifier

- A statistical classifier
 - Performs *probabilistic prediction*, i.e., predicts class membership probabilities
- Foundation
 - Based on Bayes' Theorem.
- Assumptions
 - 1.The classes are mutually exclusive and exhaustive.
 - 2.The attributes are independent given the class.
- Called “Naïve” classifier because of these assumptions.
 - Empirically proven to be useful.
 - Scales very well.

Example: Bayesian Classification

- **Example 8.2: Air Traffic Data**
 - Let us consider a set observation recorded in a database
 - Regarding the arrival of airplanes in the routes from any airport to New Delhi under certain conditions.



Air-Traffic Data

Cond. to next slide...

Air-Traffic Data

Cond. from previous slide...

Air-Traffic Data

- In this database, there are four attributes

$$A = [\text{Day}, \text{Season}, \text{Fog}, \text{Rain}]$$

with 20 tuples.

- The categories of classes are:

$$C = [\text{On Time}, \text{Late}, \text{Very Late}, \text{Cancelled}]$$

- Given this is the knowledge of data and classes, we are to find most likely classification for any other **unseen instance**, for example:

Week Day	Winter	High	None	???
----------	--------	------	------	-----

- Classification technique eventually map this tuple into an accurate class.

Bayesian Classifier

- In many applications, the relationship between the attributes set and the class variable is **non-deterministic**.
 - In other words, a test cannot be classified to a class label with certainty.
 - In such a situation, the classification can be achieved **probabilistically**.
- The Bayesian classifier is an approach for **modelling probabilistic relationships** between the attribute set and the class variable.
- More precisely, Bayesian classifier use **Bayes' Theorem of Probability** for classification.
- Before going to discuss the Bayesian classifier, we should have a quick look at the **Theory of Probability** and then **Bayes' Theorem**.

Bayes' Theorem of Probability

Simple Probability

Definition 8.2: Simple Probability

If there are n elementary events associated with a random experiment and m of n of them are favorable to an event A , then the probability of happening or occurrence of A is

Simple Probability

- Suppose, A and B are any two events and $P(A)$, $P(B)$ denote the probabilities that the events A and B will occur, respectively.
- **Mutually Exclusive Events:**
 - Two events are mutually exclusive, if the occurrence of one precludes the occurrence of the other.

Example: Tossing a coin (two events)

Tossing a ludo cube (Six events)

💡 Can you give an example, so that two events are not mutually exclusive?

Hint: Tossing two identical coins, Weather (sunny, foggy, warm)

Simple Probability

- **Independent events:** Two events are independent if occurrences of one does not alter the occurrence of other.

Example: Tossing both coin and ludo cube together.

(How many events are here?)

💡 Can you give an example, where an event is dependent on one or more other events(s)?

Hint: Receiving a message (A) through a communication channel (B) over a computer (C), rain and train.

Joint Probability

Definition 8.3: Joint Probability

If $P(A)$ and $P(B)$ are the probability of two events, then

If A and B are mutually exclusive, then

If A and B are independent events, then

Thus, for mutually exclusive events

Conditional Probability

Definition 8.2: Conditional Probability

If events are dependent, then their probability is expressed by conditional probability. The probability that A occurs given that B is denoted by .

Suppose, A and B are two events associated with a random experiment. The probability of A under the condition that B has already occurred and is given by

Conditional Probability

Corollary 8.1: Conditional Probability

or

For three events A, B and C

For n events A_1, A_2, \dots, A_n and if all events are mutually independent to each other

Note:

if events are **mutually exclusive**

if A and B are **independent**

otherwise,

Conditional Probability

- Generalization of Conditional Probability:

$$\because P(A) = P(B)$$

By the law of total probability : $P(B) =$

Conditional Probability

In general,

Total Probability

Definition 8.3: Total Probability

Let n be n mutually exclusive and exhaustive events associated with a random experiment. If A is any event which occurs with , then

Total Probability: An Example

Example 8.3

A bag contains 4 red and 3 black balls. A second bag contains 2 red and 4 black balls. One bag is selected at random. From the selected bag, one ball is drawn. What is the probability that the ball drawn is red?

This problem can be answered using the concept of Total Probability

Selecting bag I

Selecting bag II

A = Drawing the red ball

Thus,

where, $P(A|I)$ = Probability of drawing red ball when first bag has been chosen
and $P(A|II)$ = Probability of drawing red ball when second bag has been chosen

Reverse Probability

Example 8.3:

A bag (Bag I) contains 4 red and 3 black balls. A second bag (Bag II) contains 2 red and 4 black balls. You have chosen one ball at random. It is found as red ball. What is the probability that the ball is chosen from Bag I?

Here,

Selecting bag *I*

Selecting bag *II*

A = Drawing the red ball

We are to determine $P(|A)$. Such a problem can be solved using Bayes' theorem of probability.

Bayes' Theorem

Theorem 8.4: Bayes' Theorem

Let n be n mutually exclusive and exhaustive events associated with a random experiment. If A is any event which occurs with , then

Prior and Posterior Probabilities

- $P(A)$ and $P(B)$ are called prior probabilities
- $P(A|B)$, $P(B|A)$ are called posterior probabilities

Example 8.6: Prior versus Posterior Probabilities

- This table shows that the event Y has two outcomes namely A and B , which is dependent on another event X with various outcomes like and .
- **Case1:** Suppose, we don't have any information of the event A . Then, from the given sample space, we can calculate $P(Y = A) = 0.5$
- **Case2:** Now, suppose, we want to calculate $P(X = |Y = A) = 0.4$.

The later is the conditional or posterior probability, where as the former is the prior probability.

X	Y
	A
	A
	B
	A
	B
	A
	B
	B
	B
	A

Naïve Bayesian Classifier

- Suppose, Y is a class variable and $X =$ is a set of attributes, with instance of Y .

INPUT (X)	CLASS(Y)

- The classification problem, then can be expressed as the class-conditional probability

Naïve Bayesian Classifier

- Naïve Bayesian classifier calculate this posterior probability using Bayes' theorem, which is as follows.
- From Bayes' theorem on conditional probability, we have

where,

$$(Y)$$

Note:

- is called the evidence (also the total probability) and it is a constant.
- The probability $P(Y/X)$ (also called class conditional probability) is therefore proportional to $P(X/Y)$.
- Thus, $P(Y/X)$ can be taken as a measure of Y given that X .

$$P(Y/X)$$

Naïve Bayesian Classifier

- Suppose, for a given instance of X (say $x = ()$ and).
- There are any two class conditional probabilities namely $P(Y/X=x)$ and $P(YX=x)$.
- If $P(YX=x) > P(YX=x)$, then we say that is more stronger than for the instance $X = x$.
- The strongest is the classification for the instance $X = x$.

Naïve Bayesian Classifier

- **Example:** With reference to the Air Traffic Dataset mentioned earlier, let us tabulate all the posterior and prior probabilities as shown below.

		Class			
Attribute		On Time	Late	Very Late	Cancelled
Day	Weekday	9/14 = 0.64	½ = 0.5	3/3 = 1	0/1 = 0
	Saturday				
Season	Summer				
	Winter	2/14 = 0.14	2/2 = 1	2/3 = 0.67	0/1 = 0

Naïve Bayesian Classifier

		Class			
Attribute		On Time	Late	Very Late	Cancelled
Fog	High	4/14 = 0.29	1/2 = 0.5	1/3 = 0.33	1/1 = 1
	Low				
Rain	Heavy	1/14 = 0.07	1/2 = 0.5	2/3 = 0.67	1/1 = 1
	Light				
Prior Probability					

Naïve Bayesian Classifier

Instance:

Week Day	Winter	High	Heavy	???
----------	--------	------	-------	-----

Case1: Class = On Time : $0.70 \times 0.64 \times 0.14 \times 0.29 \times 0.07 = 0.0013$

Case2: Class = Late : $0.10 \times 0.50 \times 1.0 \times 0.50 \times 0.50 = 0.0125$

Case3: Class = Very Late : $0.15 \times 1.0 \times 0.67 \times 0.33 \times 0.67 = 0.0222$

Case4: Class = Cancelled : $0.05 \times 0.0 \times 0.0 \times 1.0 \times 1.0 = 0.0000$

Case3 is the strongest; Hence correct classification is **Very Late**

Naïve Bayesian Classifier

Algorithm: Naïve Bayesian Classification

Input: Given a set of k mutually exclusive and exhaustive classes $C = \{C_1, C_2, \dots, C_k\}$, which have prior probabilities $P(C_1), P(C_2), \dots, P(C_k)$.

There are n -attribute set $A = \{A_1, A_2, \dots, A_n\}$ which for a given instance have values $= v_1, = v_2, \dots, = v_n$

Step: For each C_i , calculate the class condition probabilities, $i = 1, 2, \dots, k$

Output: \hat{C}_i is the classification

Note: $P(A_i | C_j)$, because they are not probabilities rather proportion values (to posterior probabilities)
IIITS: Data Analytics

Naïve Bayesian Classifier

Pros and Cons

- The Naïve Bayes' approach is a very popular one, which often works well.
- However, it has a number of potential problems
 - It relies on all attributes being **categorical**.
 - If the data is **less**, then it **estimates poorly**.

Naïve Bayesian Classifier

Approach to overcome the limitations in Naïve Bayesian Classification

- Estimating the posterior probabilities for continuous attributes
 - In real life situation, all attributes are not necessarily be categorical, In fact, there is a mix of both categorical and continuous attributes.
 - In the following, we discuss the schemes to deal with continuous attributes in Bayesian classifier.
 1. We can discretize each continuous attributes and then replace the continuous values with its corresponding discrete intervals.
 2. We can assume a certain form of probability distribution for the continuous variable and estimate the parameters of the distribution using the training data. A Gaussian distribution is usually chosen to represent the posterior probabilities for continuous attributes. A general form of Gaussian distribution will look like

where, denote [mean](#) and [variance](#), respectively.

Naïve Bayesian Classifier

For each class C_i , the posterior probabilities for attribute A_j (it is the numeric attribute) can be calculated following Gaussian normal distribution as follows.

Here, the parameter μ_{ij} can be calculated based on the sample mean of attribute value of A_j for the training records that belong to the class C_i .

Similarly, σ_{ij}^2 can be estimated from the calculation of variance of such training records.

Naïve Bayesian Classifier

M-estimate of Conditional Probability

- The M-estimation is to deal with the potential problem of Naïve Bayesian Classifier when training data size is too poor.
 - If the posterior probability for one of the attribute is zero, then the overall class-conditional probability for the class vanishes.
 - In other words, if training data do not cover many of the attribute values, then we may not be able to classify some of the test records.
- This problem can be addressed by using the M-estimate approach.

M-estimate Approach

- M-estimate approach can be stated as follows

where, n = total number of instances from class

= number of training examples from class that take the value

m = it is a parameter known as the equivalent sample size, and

p = is a user specified parameter.

Note:

If $n = 0$, that is, if there is no training set available, then $= p$,
so, this is a different value, in absence of sample value.

A Practice Example

Example 8.4

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data instance

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

A Practice Example

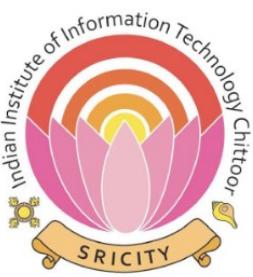
- $P(C_i)$:
 $P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$
 $P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$
- Compute $P(X|C_i)$ for each class
 - $P(\text{age} = \text{"<=30"} | \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$
 - $P(\text{age} = \text{"<= 30"} | \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$

$$P(X|C_i) : P(X|\text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$
$$P(X|\text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i)*P(C_i) : P(X|\text{buys_computer} = \text{"yes"}) * P(\text{buys_computer} = \text{"yes"}) = 0.028$$
$$P(X|\text{buys_computer} = \text{"no"}) * P(\text{buys_computer} = \text{"no"}) = 0.007$$

Therefore, X belongs to class ("buys_computer = yes")

Thank You!



Brain Computer Interaction

Feature Translation – Decision Trees

Course Instructors

Dr. Annushree Bablani

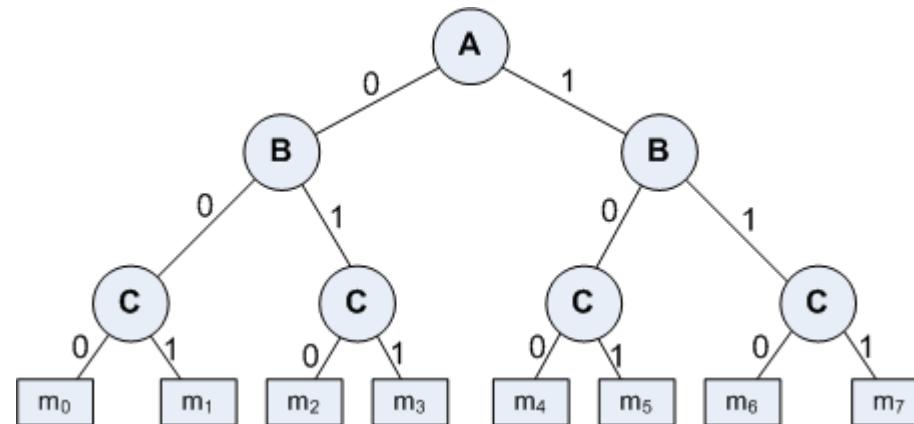
Acknowledgements: Dr. Sreeja S R

Basic Concept

- A Decision Tree is an important data structure known to solve many computational problems

Example 8.1: Binary Decision Tree

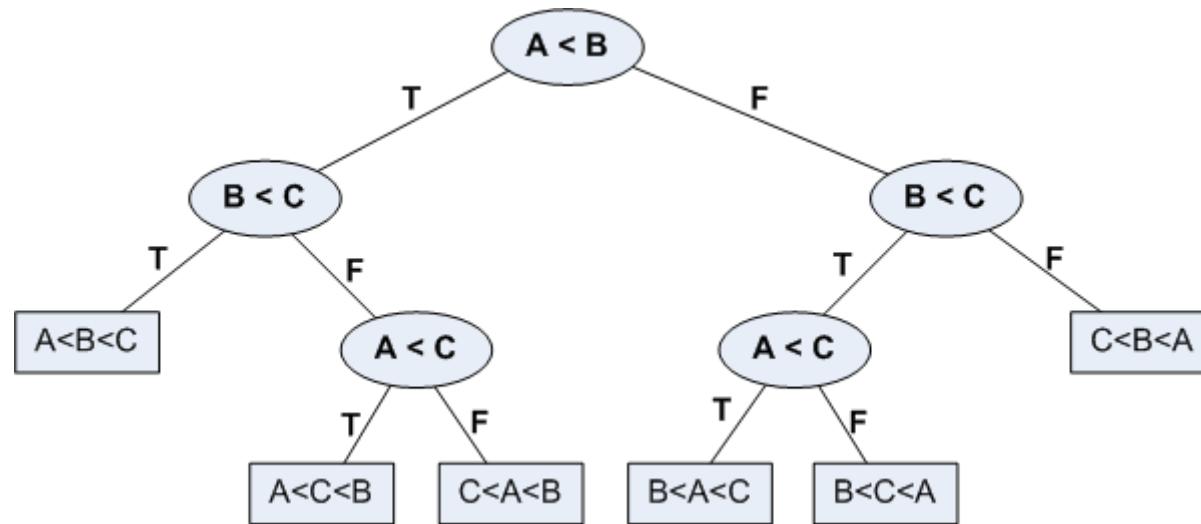
A	B	C	f
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7



Basic Concept

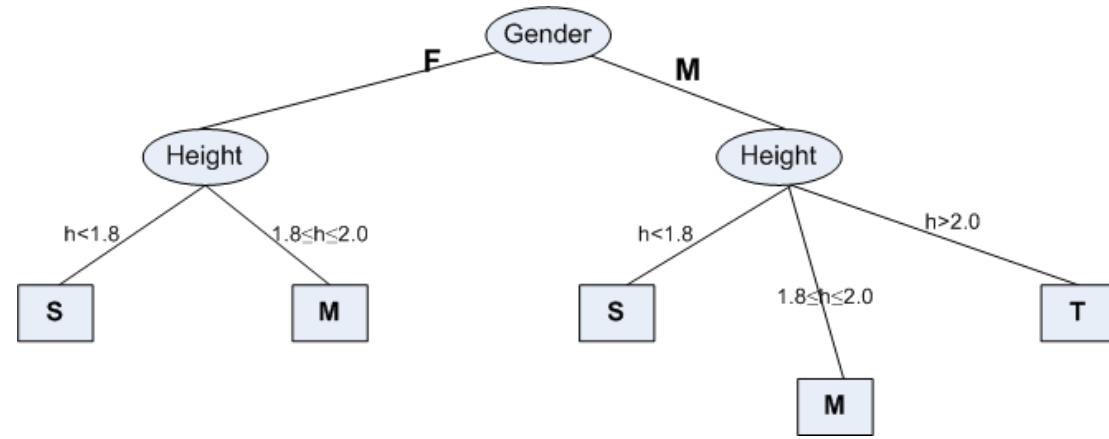
- In Example 18.1, we have considered a decision tree where values of any attribute if binary only. Decision tree is also possible where attributes are of continuous data type

Example 8.2: Decision Tree with numeric data



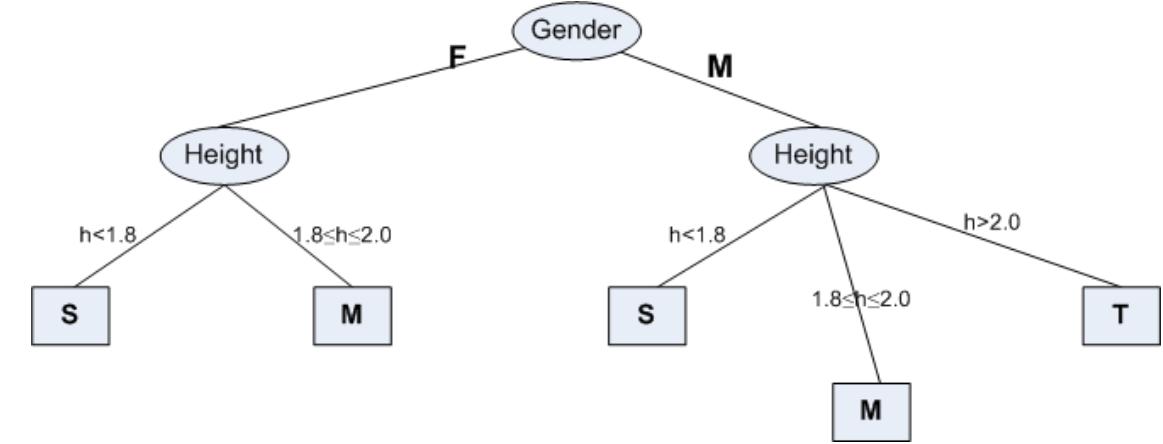
Some Characteristics

- Decision tree may be n -ary, $n \geq 2$.
- There is a special node called **root node**.
- All nodes drawn with circle (ellipse) are called **internal nodes**.
- All nodes drawn with rectangle boxes are called **terminal nodes** or **leaf nodes**.
- Edges of a node represent the **outcome for a value** of the node.
- In a path, a node with same label **is never repeated**.
- Decision tree **is not unique**, as different ordering of internal nodes can give different decision tree.



Decision Tree and Classification Task

- Decision tree helps us to classify data.
 - Internal nodes are some attribute
 - Edges are the values of attributes
 - External nodes are the outcome of classification
- Such a classification is, in fact, made by posing questions starting from the root node to each terminal node.



Decision Tree and Classification Task

Example 8.3 : Vertebrate Classification

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	Mammal
Python	Cold	scales	no	no	no	no	yes	Reptile
Salmon	Cold	scales	no	yes	no	no	no	Fish
Whale	Warm	hair	yes	yes	no	no	no	Mammal
Frog	Cold	none	no	semi	no	yes	yes	Amphibian
Komodo	Cold	scales	no	no	no	yes	no	Reptile
Bat	Warm	hair	yes	no	yes	yes	yes	Mammal
Pigeon	Warm	feathers	no	no	yes	yes	no	Bird
Cat	Warm	fur	yes	no	no	yes	no	Mammal
Leopard	Cold	scales	yes	yes	no	no	no	Fish
Turtle	Cold	scales	no	semi	no	yes	no	Reptile
Penguin	Warm	feathers	no	semi	no	yes	no	Bird
Porcupine	Warm	quills	yes	no	no	yes	yes	Mammal
Eel	Cold	scales	no	yes	no	no	no	Fish
Salamander	Cold	none	no	semi	no	yes	yes	Amphibian

What are the class label of Dragon and Shark?

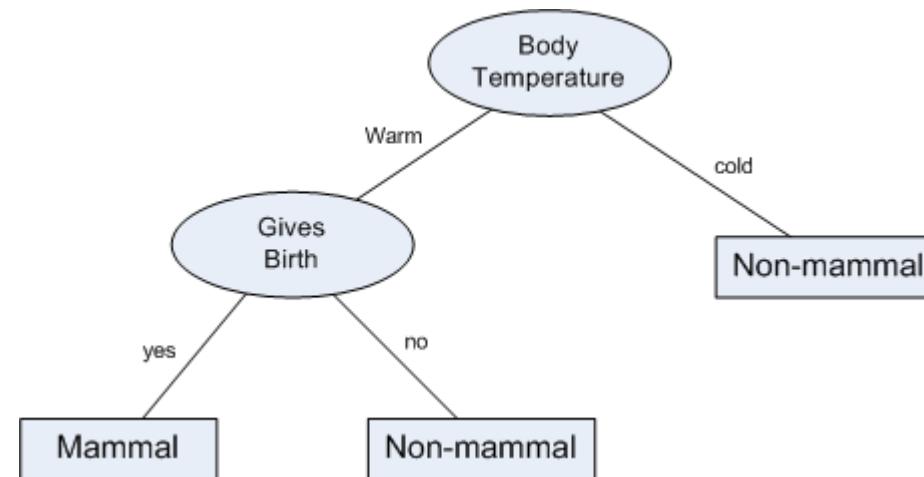
Decision Tree and Classification Task

Example 8.3 : Vertebrate Classification

- Suppose, a new species is discovered as follows.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Gila Monster	cold	scale	no	no	no	yes	yes	?

- Decision Tree that can be induced based on the data (in Example 8.3) is as follows.



Decision Tree and Classification Task

- The above Example illustrates how we can solve a classification problem by asking a series of question about the attributes.
 - Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class-label of the test.
- The series of questions and their answers can be organized in the form of a decision tree
 - As a hierarchical structure consisting of nodes and edges
 - Once a decision tree is built, it is applied to any test to classify it.

Definition of Decision Tree

Definition 1: Decision Tree

Given a database D = here denotes a tuple, which is defined by a set of attribute set of classes C = .

A decision tree T is a tree associated with D that has the following properties:

- Each internal node is labeled with an attribute A_i
- Each edges is labeled with predicate that can be applied to the attribute associated with the parent node of it
- Each leaf node is labeled with class c_j

Building Decision Tree

- In principle, there are exponentially many decision tree that can be constructed from a given database (also called training data).
 - Some of the tree may not be optimum
 - Some of them may give inaccurate result
- Two approaches are known
 - **Greedy strategy**
 - A top-down recursive divide-and-conquer
 - **Modification of greedy strategy**
 - ID3
 - C4.5
 - CART, etc.

Node Splitting in BuildDT Algorithm

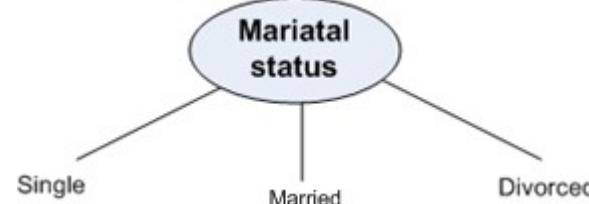
- BuildDT algorithm must provides a method for expressing **an attribute test condition** and **corresponding outcome** for different attribute type
- **Case: Binary attribute**
 - This is the simplest case of node splitting
 - The test condition for a binary attribute generates only two outcomes



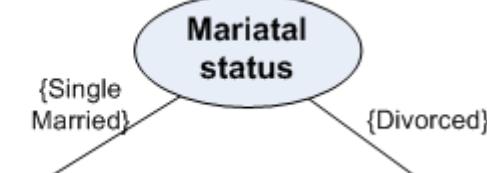
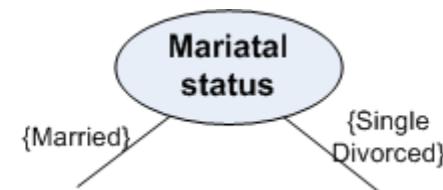
Node Splitting in BuildDT Algorithm

- **Case: Nominal attribute**

- Since a nominal attribute can have many values, its test condition can be expressed in two ways:
 - A multi-way split
 - A binary split
- **Muti-way split:** Outcome depends on the number of distinct values for the corresponding attribute

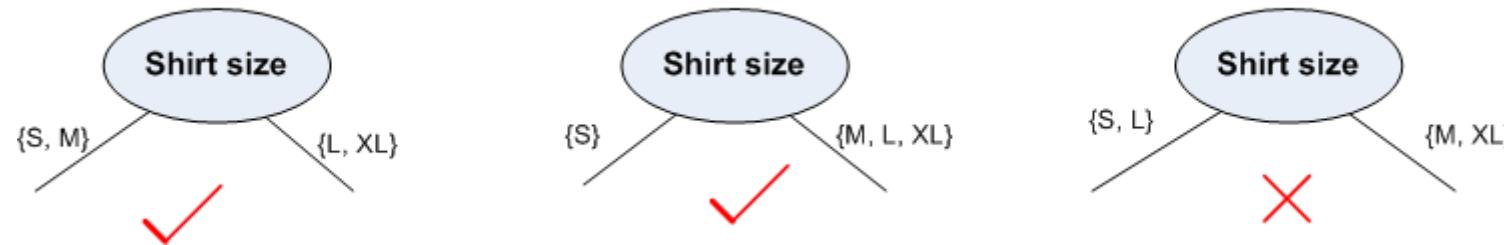


- **Binary splitting** by grouping attribute values



Node Splitting in BuildDT Algorithm

- **Case: Ordinal attribute**
 - It also can be expressed in two ways:
 - A multi-way split
 - A binary split
 - **Multi-way split:** It is same as in the case of nominal attribute
 - **Binary splitting** attribute values should be grouped maintaining the **order** property of the attribute values



Node Splitting in BuildDT Algorithm

- **Case: Numerical attribute**

- For numeric attribute (with discrete or continuous values), a test condition can be expressed as a comparison set
 - **Binary outcome:** $A > v$ or $A \leq v$
 - In this case, decision tree induction must consider all possible split positions
 - **Range query :** $v_i \leq A < v_{i+1}$ for $i = 1, 2, \dots, q$ (if q number of ranges are chosen)
 - Here, q should be decided a priori
- For a numeric attribute, decision tree induction is a combinatorial optimization problem

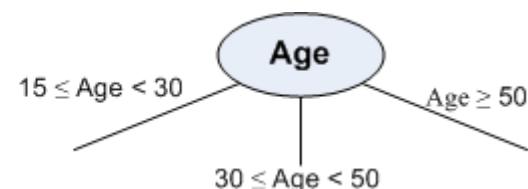
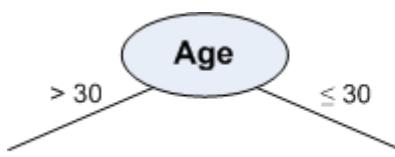


Illustration : BuildDT Algorithm

Example 8.4: Illustration of BuildDT Algorithm

- Consider a training data set as shown.

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

Attributes:

Gender = {Male(M), Female (F)} // Binary attribute

Height = {1.5, ..., 2.5} // Continuous attribute

Class = {Short (S), Medium (M), Tall (T)}

Given a person, we are to test in which class s/he belongs

Illustration : BuildDT Algorithm

- To built a decision tree, we can select an attribute in two different orderings: $\langle \text{Gender}, \text{Height} \rangle$ or $\langle \text{Height}, \text{Gender} \rangle$
- Further, for each ordering, we can choose different ways of splitting
- Different instances are shown in the following.
- **Approach 1 : $\langle \text{Gender}, \text{Height} \rangle$**

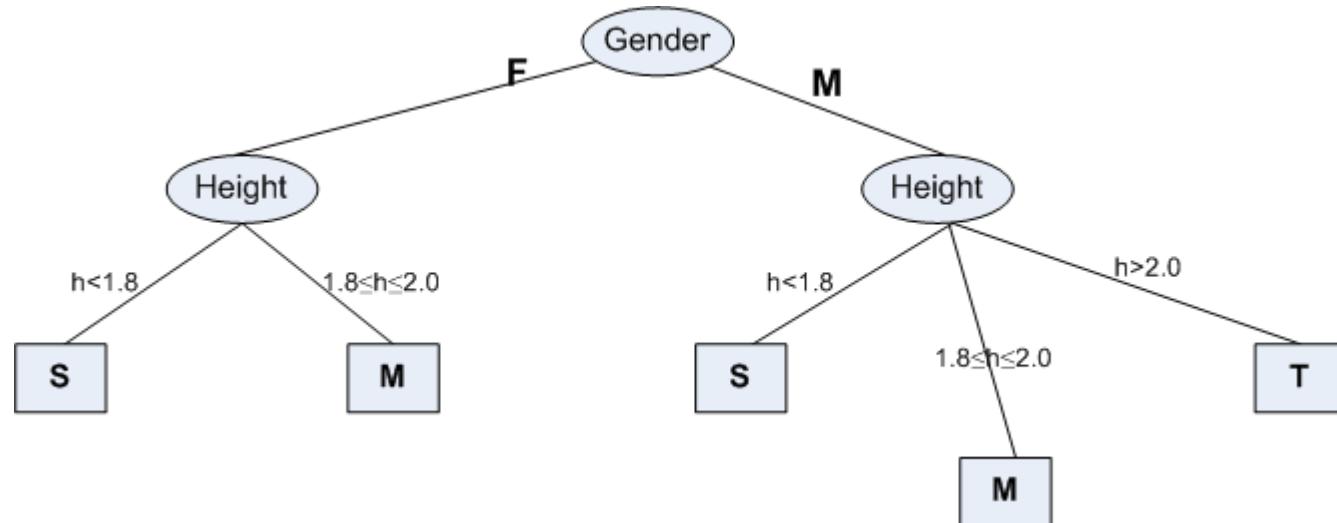


Illustration : BuildDT Algorithm

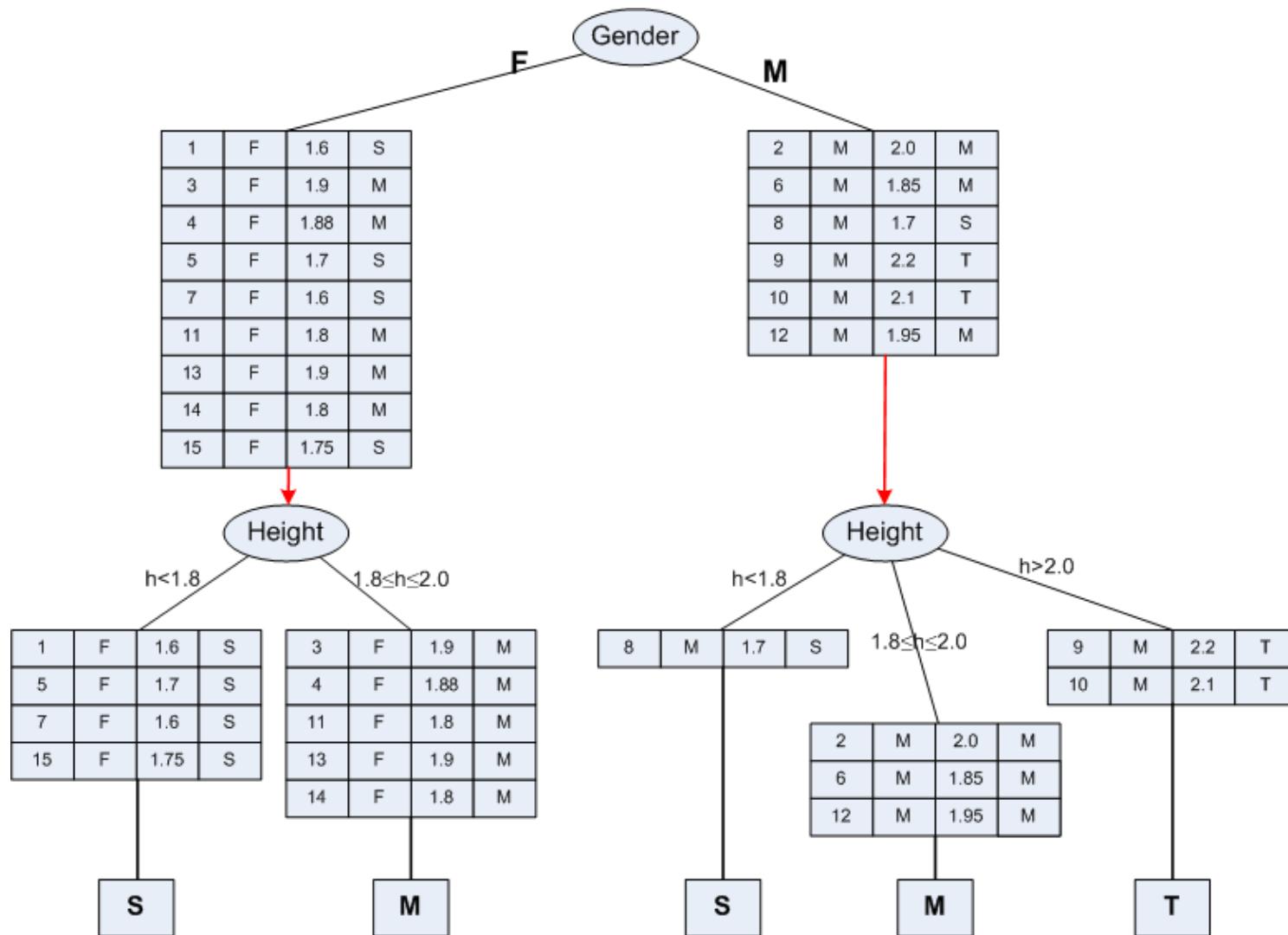


Illustration : BuildDT Algorithm

- Approach 2 : <Height, Gender>

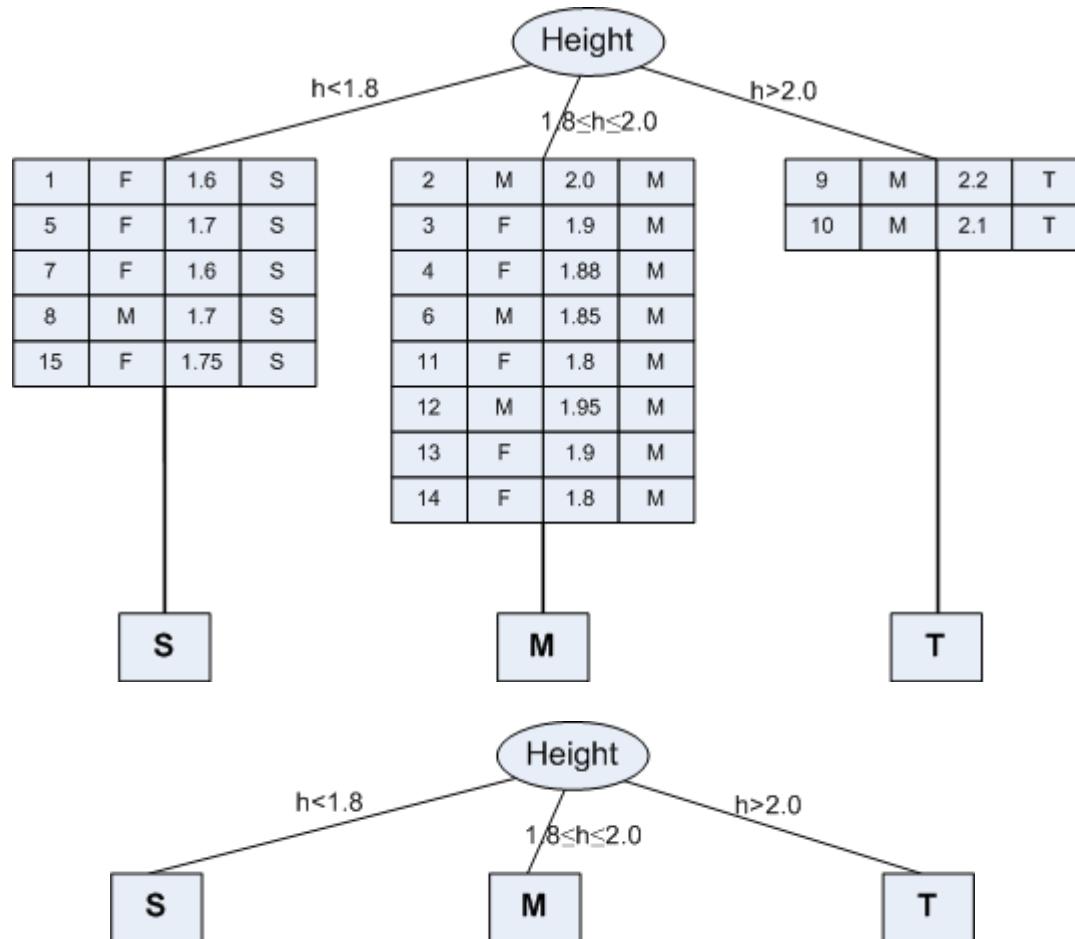


Illustration : BuildDT Algorithm

Example 8.5: Illustration of BuildDT Algorithm

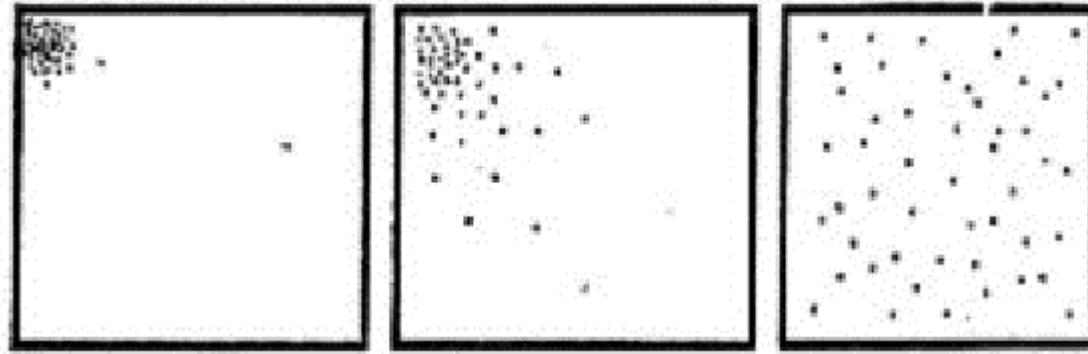
- Consider an anonymous database as shown.

A1	A2	A3	A4	Class
a11	a21	a31	a41	C1
a12	a21	a31	a42	C1
a11	a21	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a31	a42	C1
a11	a21	a32	a42	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a12	a22	a31	a42	C1

- Is there any “clue” that enables to select the “best” attribute first?
- Suppose, following are two attempts:
 - $A1 \rightarrow A2 \rightarrow A3 \rightarrow A4$ [naïve]
 - $A3 \rightarrow A2 \rightarrow A4 \rightarrow A1$ [Random]
- Draw the decision trees in the above-mentioned two cases.
- Are the trees different to classify any test data?
- If any other sample data is added into the database, is that likely to alter the decision tree already obtained?

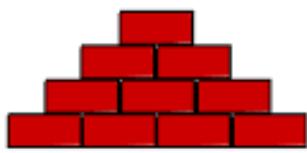
Concept of Entropy

Concept of Entropy

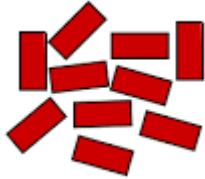


If a point represents a gas molecule,
then which system has the more
entropy?

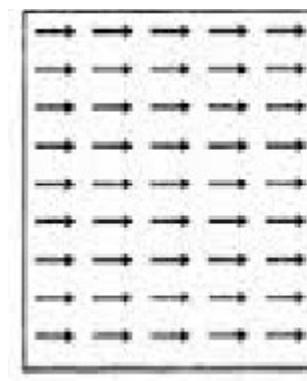
How to measure? ?



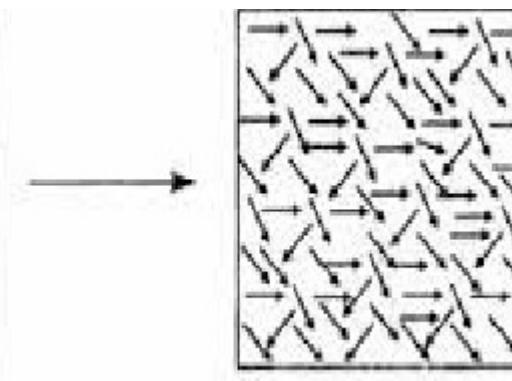
More ordered
less entropy



Less ordered
higher entropy



More organized or
ordered (less probable)



Less organized or
disordered (more probable)

An Open Challenge!

Roll No.	Assignment	Project	Mid-Sem	End-Sem
12BT3FP06	89	99	56	91
10IM30013	95	98	55	93
12CE31005	98	96	58	97
12EC35015	93	95	54	99
12GG2005	90	91	53	98
12MI33006	91	93	57	97
13AG36001	96	94	58	95
13EE10009	92	96	56	96
13MA20012	88	98	59	96
14CS30017	94	90	60	94
14ME10067	90	92	58	95
14MT10038	99	89	55	93

Roll No.	Assignment	Project	Mid-Sem	End-Sem
12BT3FP06	19	59	16	71
10IM30013	37	38	25	83
12CE31005	38	16	48	97
12EC35015	23	95	54	19
12GG2005	40	71	43	28
12MI33006	61	93	47	97
13AG36001	26	64	48	75
13EE10009	92	46	56	56
13MA20012	88	58	59	66
14CS30017	74	20	60	44
14ME10067	50	42	38	35
14MT10038	29	69	25	33

Two sheets showing the tabulation of marks obtained in a course are shown.

Which tabulation of marks shows the “good” performance of the class?

How you can measure the same?

Entropy and its Meaning

- Entropy is an important concept used in Physics in the context of heat and thereby uncertainty of the states of a matter.
- At a later stage, with the growth of Information Technology, entropy becomes an important concept in [Information Theory](#).
- To deal with the classification job, entropy is an important concept, which is considered as
 - [an information-theoretic measure of the “uncertainty” contained in a training data](#)
 - [due to the presence of more than one classes.](#)

Entropy in Information Theory

- The entropy concept in information theory first time coined by Claude Shannon (1850).
- The first time it was used to measure the “information content” in messages.
- According to his concept of entropy, presently entropy is widely being used as a way of representing messages for efficient transmission by Telecommunication Systems.

Measure of Information Content

- People, in general, are information hungry!
- Everybody wants to acquire information (from newspaper, library, nature, fellows, etc.)
 - Think how a crime detector do it to know about the crime from crime spot and criminal(s).
 - Kids annoyed their parents asking questions.
- In fact, fundamental thing is that we gather information asking questions (and decision tree induction is no exception).
 - We may note that information gathering may be with certainty or uncertainty.

Measure of Information Content

Example 8.6

- a) Guessing a birthday of your classmate

It is with uncertainty ~

Whereas guessing the day of his/her birthday is .

This uncertainty, we may say varies between 0 to 1, both inclusive.

- b) As another example, a question related to event with eventuality (or impossibility) will be answered with 0 or 1 uncertainty.

• Does sun rises in the East? (answer is with 0 uncertainty)

• Will mother give birth to male baby? (answer is with $\frac{1}{2}$ uncertainty)

• Is there a planet like earth in the galaxy? (answer is with an extreme uncertainty)

Entropy Calculation

- If there are m objects with frequencies , , then the average number of bits (i.e. questions) that need to be examined a value, that is, entropy is the frequency of occurrence of the value multiplied by the number of bits that need to be determined, summed up values of from l to m .

Theorem: Entropy calculation

If p_i denotes the frequencies of occurrences of m distinct objects, then the entropy E is

Note:

- If all are equally likely, then and ; it is the special case.

Entropy of a Training Set

- If there are k classes , , and for denotes the number of occurrences of classes divided by the total number of instances (i.e., the frequency of occurrence of) in the training set, then entropy of the training set is denoted by

Here, E is measured in “bits” of information.

Note:

- The above formula should be summed over the non-empty classes only, that is, classes for which
- E is always a positive quantity
- E takes it's minimum value (zero) if and only if all the instances have the same class (i.e., the training set with only one non-empty class, for which the probability 1).
- Entropy takes its maximum value when the instances are equally distributed among k possible classes. In this case, the maximum value of E is .

Entropy of a Training Set

Example 18.10: OPTH dataset

Consider the OTPH data shown in the following table with total 24 instances in it.

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
2	1	2	1	3
2	1	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	2	2	2	1
2	2	1	1	3
2	2	1	2	2
3	1	1	1	3
3	1	1	2	2
3	2	2	1	3
3	2	2	2	2
3	2	1	1	3
3	2	1	2	2

A coded forms for all values of attributes are used to avoid the cluttering in the table.

Entropy of a training set

Specification of the attributes are as follows.

Age	Eye Sight	Astigmatic	Use Type
1: Young	1: Myopia	1: No	1: Frequent
2: Middle-aged	2: Hypermetropia	2: Yes	2: Less
3: Old			

Class: 1: Contact Lens 2:Normal glass 3: Nothing

In the OPTH database, there are 3 classes and 4 instances with class 1, 5 instances with class 2 and 15 instances with class 3. Hence, entropy E of the database is:

Note:

- The entropy of a training set implies the number of yes/no questions, on the average, needed to determine an unknown test to be classified.
- It is very crucial to decide the series of questions about the value of a set of attribute, which collectively determine the classification. Sometimes it may take one question, sometimes many more.
- Decision tree induction helps us to ask such a series of questions. In other words, we can utilize entropy concept to build a better decision tree.

How entropy can be used to build a decision tree ?

Decision Tree Induction Techniques

- Decision tree induction is a top-down, recursive and divide-and-conquer approach.
- The procedure is to choose an attribute and split it into from a larger training set into smaller training sets.
- Different algorithms have been proposed to take a good control over
 1. Choosing the best attribute to be splitted, and
 2. Splitting criteria
- Several algorithms have been proposed for the above tasks. In this lecture, we shall limit our discussions into three important of them
 - **ID3**
 - **C 4.5**
 - **CART**

Algorithm ID3

ID3: Decision Tree Induction Algorithms

- Quinlan [1984] introduced the ID3, a popular short form of Iterative Dichotomizer 3 for decision trees from a set of training data.
- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.
- At each node, the splitting attribute is selected to be the most informative among the attributes not yet considered in the path starting from the root.

Algorithm ID3

- In ID3, **entropy is used** to measure how informative a node is.
 - It is observed that splitting on any attribute has **the property that average entropy of the resulting training subsets will be less than or equal to** that of the previous training set.
- ID3 algorithm defines a measurement of a splitting called **Information Gain** to determine the goodness of a split.
 - The attribute with the **largest value of information gain** is chosen as the splitting attribute and
 - it partitions into a number of smaller training sets based on the **distinct values of attribute** under split.

Defining Information Gain

- We consider the following symbols and terminologies to define information gain, which is denoted as α .
- D denotes the training set at any instant
- $|D|$ denotes the size of the training set D
- $E(D)$ denotes the entropy of the training set D
- The entropy of the training set D

$$E(D) = -$$

- where the training set D has , , . . . , , the k number of distinct classes and
- , $0 < p_i \leq 1$ is the probability that an arbitrary tuple in D belongs to class $(i = 1, 2, \dots, k)$.

Defining Information Gain

Definition 2: Weighted Entropy

The weighted entropy denoted as $E_A(D)$ for all partitions of D with respect to A is given by:

$$= E()$$

Here, the term w_j denotes the weight of the j -th training set.

More meaningfully, $E_A(D)$ is the expected information required to classify a tuple from D based on the splitting of A .

Defining Information Gain

- Our objective is to take A on splitting to produce an exact classification (also called pure), that is, all tuples belong to one class.
- However, it is quite likely that the partitions is impure, that is, they contain tuples from two or more classes.
- In that sense, $I(A)$ is a measure of impurities (or purity). A lesser value of $I(A)$ implying more power the partitions are.

Definition 3: Information Gain

Information gain, $I(A)$, of the training set D splitting on the attribute A is given by
$$I(A) = E(D) - \sum_{j=1}^{n_A} p_j E(D_j)$$

In other words, $I(A)$ gives us an estimation how much would be gained by splitting on A . The attribute A with the highest value of $I(A)$ should be chosen as the splitting attribute for D .

Information Gain Calculation

Example 8.11 : Information gain on splitting OPTH

- Let us refer to the OPTH database discussed earlier.
- Splitting on **Age** at the root level, it would give three subsets and as shown in the tables in the following three slides.
- The entropy and of training sets and and corresponding weighted entropy and are also shown alongside.
- The Information gain is then can be calculated as **0.0394**.
- Recall that entropy of OPTH data set, we have calculated as $E(OPTH) = 1.3261$ (*see Slide #16*)

Information Gain Calculation

Example 8.11 : Information gain on splitting OPTH

Training set: (Age = 1)

Age	Eye-sight	Astigmatism	Use type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1

$$0.0 \times 1.5 = 0.5000$$

Calculating Information Gain

Training set: (Age = 2)

Age	Eye-sight	Astigmatism	Use type	Class
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3

$$= 1.2988$$

$$\times 1.2988 = 0.4329$$

Calculating Information Gain

Training set: (Age = 3)

Age	Eye-sight	Astigmatism	Use type	Class
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

(0)

$$0 = 1.0613$$

$$\times 1.0613 = 0.3504$$

$$= 1.3261 - (0.5000 + 0.4329 + 0.3504) = \mathbf{0.0394}$$

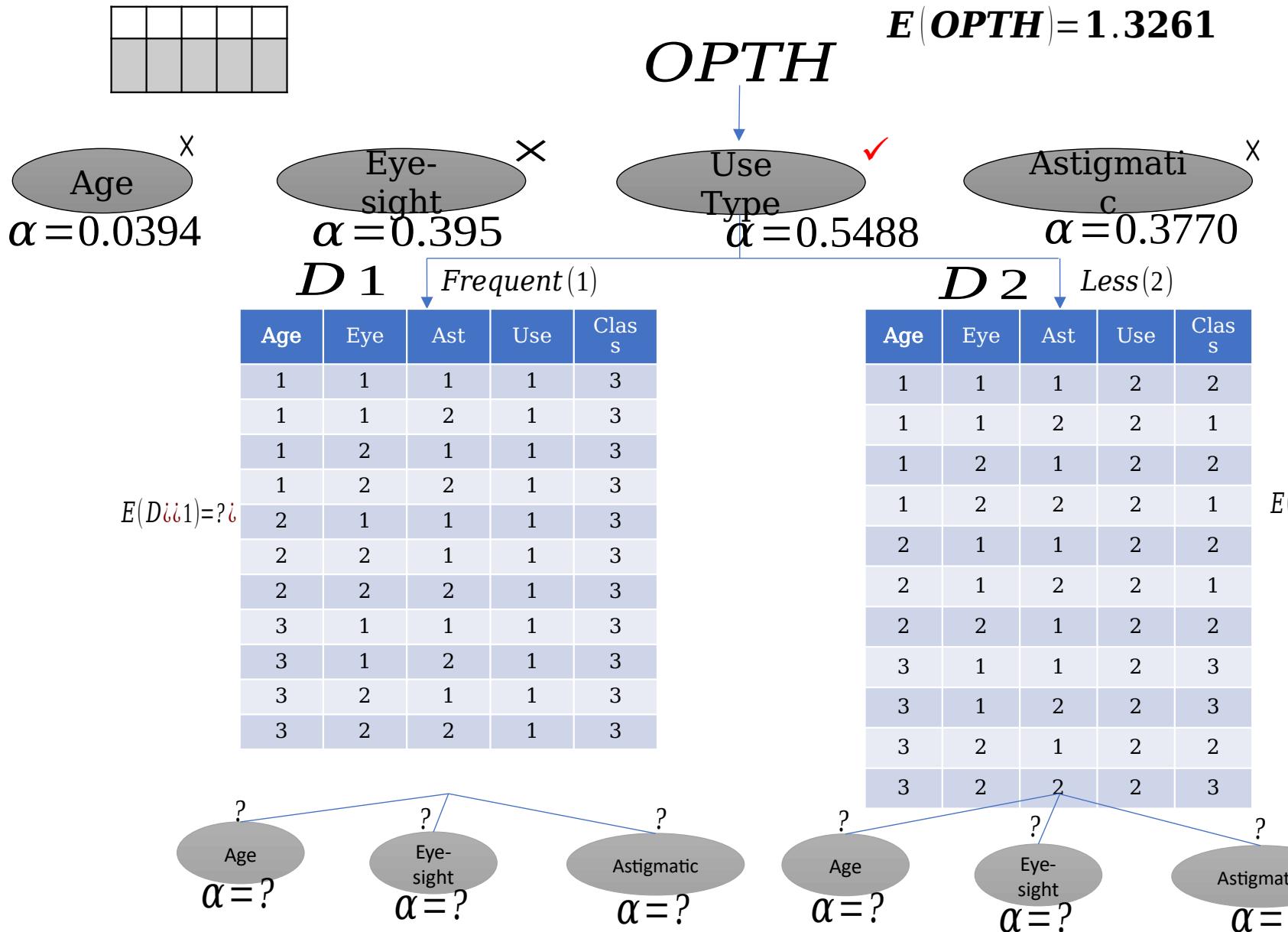
Information Gains for Different Attributes

- In the same way, we can calculate the information gains, when splitting the OPTH database on Eye-sight, Astigmatic and Use Type. The results are summarized below.
- Splitting attribute: Age
- Splitting attribute: Eye-sight
- Splitting attribute: Astigmatic
770
- Splitting attribute: Use Type
5488

Decision Tree Induction : ID3 Way

- The ID3 strategy of attribute selection is to choose to split on the attribute that gives the greatest reduction in the weighted average entropy
 - The one that maximizes the value of information gain
- In the example with OPTH database, the larger values of information gain is 5488
 - Hence, the attribute should be chosen for splitting is “[Use Type](#)”.
- The process of splitting on nodes is repeated for each branch of the evolving decision tree, and the final tree, which would look like is shown in the following slide and calculation is left for practice.

Decision Tree Induction : ID3 Way



Frequency Table : Calculating α

- Calculation of entropy for each table and hence information gain for a particular split appears tedious (at least manually)!
- As an alternative, we discuss **a short-cut method** of doing the same using a special data structure called **Frequency Table**.
- **Frequency Table:** Suppose, denotes an attribute with attribute values in it. For a given database , there are a set of say . Given this, a frequency table will look like as follows.

Frequency Table : Calculating α

			X			
			
			
⋮	⋮	⋮	⋮	⋮
			
⋮	⋮	⋮	⋮	⋮

- Number of rows = Number of classes
- Number of columns = Number of attribute values
- $=$ Frequency of X for class

Assume that , the number of total instances of .

Calculation of α using Frequency Table

Example 8.12 : OTPH Dataset

With reference to OPTH dataset, and for the attribute Age, the frequency table would look like

	Age=1	Age=2	Age=3	Row Sum
Class 1	2	1	1	4
Class 2	2	2	1	5
Class 3	4	5	6	15
Column Sum	8	8	8	24

N=24

Calculation of α using Frequency Table

- The weighted average entropy then can be calculated from the frequency table following the
 - Calculate for all
 $(Entry\ Sum)$ and
 - Calculate for all
 $(Column\ Sum)$ in the row of column sum
 - Calculate

Example 8.13: OTPH Dataset

For the frequency table in Example 18.12, we have

Limiting Values of Information Gain

- The Information gain metric used in ID3 always should be positive or zero.
- It is always positive value because information is always gained (i.e., purity is improved) by splitting on an attribute.
- On the other hand, when a training set is such that if there are classes, and the entropy of training set takes the largest value i.e., (this occurs when the classes are balanced), then the information gain will be zero.

Limiting Values of Information Gain

Example 8.14: Limiting values of Information gain

Consider a training set shown below.

Data set <i>Table A</i>		
X	Y	Class
1	1	A
1	2	B
2	1	A
2	2	B
3	2	A
3	1	B
4	2	A
4	1	B

X <i>Table X</i>				
	1	2	3	4
A	1	1	1	1
B	1	1	1	1
	2	2	2	2

Frequency table of X

Y <i>Table Y</i>		
	1	2
A	2	2
B	2	2
	4	4

Frequency table of Y

Limits of Information Gain

- Entropy of Table A is (The maximum entropy).
- In this example, whichever attribute is chosen for splitting, each of the branches will also be balanced thus each with maximum entropy.
- In other words, information gain in both cases (i.e., splitting on X as well as Y) will be zero.

Note:

- The absence of information gain does not imply that there is no profit for splitting on the attribute.
- Even if it is chosen for splitting, ultimately it will lead to a final decision tree with the branches terminated by a leaf node and thus having an entropy of zero.
- Information gain can never be a negative value.

Reference

- The detail material related to this lecture can be found in

Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2015.

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison-Wesley, 2014

Any question?



Brain Computer Interaction

Feature Translation – Decision Tree Induction – CART & C4.5

Course Instructors

Dr. Annushree Bablani

Acknowledgements: Dr. Sreeja S R

Algorithm CART

CART Algorithm

- It is observed that information gain measure used in ID3 **is biased towards test with many outcomes**, that is, it prefers to select attributes having a large number of values.
- L. Breiman, J. Friedman, R. Olshen and C. Stone in 1984 proposed an algorithm to build a binary decision tree also called CART decision tree.
 - CART stands for **Classification and Regression Tree**
 - In fact, invented independently at the same time as ID3 (1984).
 - ID3 and CART are two cornerstone algorithms spawned a flurry of work on decision tree induction.
- CART is a technique that generates **a binary decision tree**; That is, unlike ID3, in CART, for each node only two children are created.
- ID3 uses Information gain as a measure to select the best attribute to be splitted, whereas CART do the same but using another measurement called **Gini index**. It is also known as **Gini Index of Diversity** and is denoted as .

Gini Index of Diversity

Definition 20.1: Gini Index

Suppose, D is a training set with size $|D|$ and C be the set of k classifications and A be any attribute with m different values of it. Like entropy measure in ID3, CART proposes Gini Index (denoted by G) as the measure of impurity of D . It can be defined as follows.

where p_{c_j} is the probability that a tuple in D belongs to class c_j and n_{c_j} can be estimated as

where n_{c_j} denotes the number of tuples in D with class c_j .

Gini Index of Diversity

Note

- measures the “impurity” of data set D .
- The **smallest value** of is zero
 - which it takes when all the classifications are same.
- It takes its **largest value**
 - when the classes are evenly distributed between the tuples, that is the frequency of each class is .

Gini Index of Diversity

Definition 20.2: Gini Index of Diversity

Suppose, a binary partition on A splits D into D_1 and D_2 , then the **weighted average Gini Index of splitting** denoted by $G(A)$ is given by

This binary partition of D reduces the impurity and the reduction in impurity is measured by

Gini Index of Diversity and CART

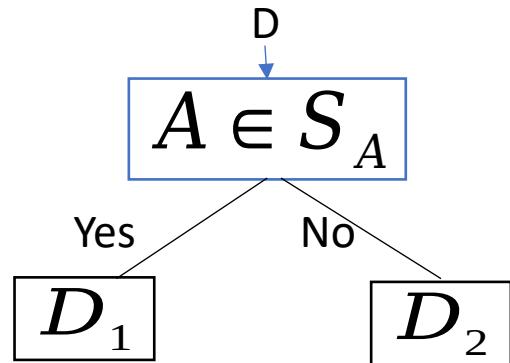
- This is called the Gini Index of diversity.
- It is also called as “impurity reduction”.
- The attribute that **maximizes** the reduction in impurity (or equivalently, has the **minimum value of**) is selected for the attribute to be splitted.

n-ary Attribute Values to Binary Splitting

- The CART algorithm considers a binary split for each attribute.
- We shall discuss how the same is possible for attribute with more than two values.
- **Case 1: Discrete valued attributes**
 - Let us consider the case where A is a discrete-valued attribute having m discrete values .
 - To determine the best binary split on A , we examine all of the possible subsets say of A that can be formed using the values of A .
 - Each subset can be considered as a binary test for attribute A of the form .

n-ary Attribute Values to Binary Splitting

- Thus, given a data set D , we have to perform a test for an attribute value A like



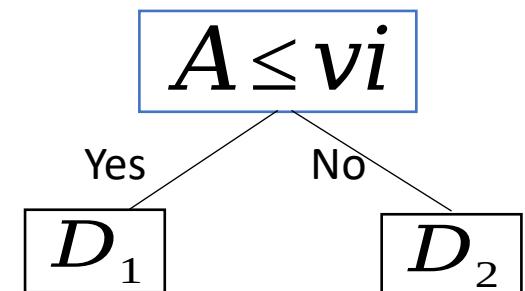
- This test is satisfied if the value of A for the tuples is among the values listed in S_A .
- If A has m distinct values in D , then there are 2^m possible subsets, out of which the empty subset and the power set should be excluded (as they really do not represent a split).
- Thus, there are $2^m - 2$ possible ways to form two partitions of the dataset D , based on the binary split of A .

n-ary Attribute Values to Binary Splitting

Case2: Continuous valued attributes

- For a continuous-valued attribute, each possible split point must be taken into account.
- According to that strategy, the mid-point between a_i and a_{i+1} , let it be v_i , then

$$v_i = \frac{a_i + a_{i+1}}{2}$$



n-ary Attribute Values to Binary Splitting

- Each pair of (sorted) adjacent values is taken as a possible split-point say.
- $\{x \in D : x \leq v\}$ is the set of tuples in D satisfying $x \leq v$ and $\{x \in D : x > v\}$ is the set of tuples in D satisfying $x > v$.
- The point giving the **minimum Gini Index** is taken as the split-point of the attribute A .

Note

- The attribute A and either its splitting subset (for discrete-valued splitting attribute) or split-point (for continuous valued splitting attribute) together form the splitting criteria.

CART Algorithm : Illustration

Example 20.1 : CART Algorithm

Suppose we want to build decision tree for the data set EMP as given in the table below.

Age

Y : young

M : middle-aged

O : old

Salary

L : low

M : medium

H : high

Job

G : government

P : private

Performance

A : Average

E : Excellent

Class : Select

Y : yes

N : no

Tuple#	Age	Salary	Job	Performance	Select
1	Y	H	P	A	N
2	Y	H	P	E	N
3	M	H	P	A	Y
4	O	M	P	A	Y
5	O	L	G	A	Y
6	O	L	G	E	N
7	M	L	G	E	Y
8	Y	M	P	A	N
9	Y	L	G	A	Y
10	O	M	G	A	Y
11	Y	M	G	E	Y
12	M	M	P	E	Y
13	M	H	G	A	Y
14	O	M	P	E	N

CART Algorithm : Illustration

For the EMP data set,

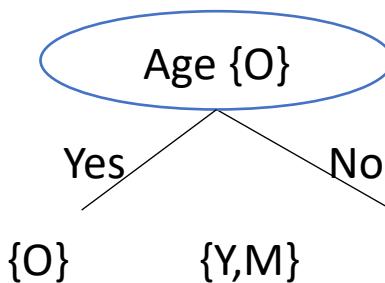
Now let us consider the calculation of $Gini$ for **Age, Salary, Job and Performance**.

CART Algorithm : Illustration

Attribute of splitting: Age

The attribute age has three values, namely Y, M and O. So there are 6 subsets, that should be considered for splitting as:

?
?

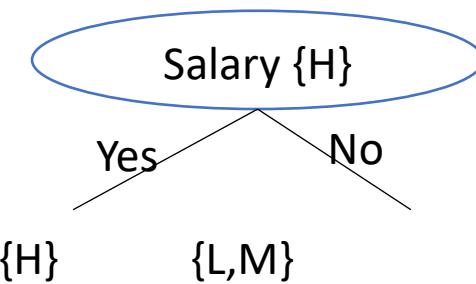


The best value of Gini Index while splitting attribute Age is

CART Algorithm : Illustration

Attribute of Splitting: Salary

The attribute salary has three values namely L , M and H . So, there are 6 subsets, that should be considered for splitting as:



CART Algorithm : Illustration

Attribute of Splitting: job

Job being the binary attribute , we have

?

?

CART Algorithm : Illustration

Attribute of Splitting: Performance

Job being the binary attribute , we have

?

?

Out of these gives the minimum value and hence, the attribute **Salary** would be chosen for splitting subset or .

Note:

It can be noted that the procedure following “information gain” calculation (i.e.) and that of “impurity reduction” calculation (i.e.) are near about.

Calculating γ using Frequency Table

- We have learnt that splitting on an attribute gives a reduction in the average Gini Index of the resulting subsets (as it does for entropy).
- Thus, in the same way the average weighted Gini Index can be calculated using the same frequency table used to calculate information gain which is as follows.

The $G()$ for the subset

Calculating γ using Frequency Table

The average weighted Gini Index, (γ) (assuming that attribute has m distinct values is)

The above gives a formula for m -attribute values; however, it can be fine tuned to subset of attributes also.

Illustration: Calculating γ using Frequency Table

Example 20.2 : Calculating γ using frequency table of OPTH

Let us consider the frequency table for OPTH database considered earlier. Also consider the attribute with three values 1, 2 and 3. The frequency table is shown below.

	1	2	3	Row sum
Class 1	2	1	1	4
Class 2	2	2	1	5
Class 3	4	5	6	15
Column sum	8	8	8	24

Illustration: Calculating γ using Frequency Table

Now we can calculate the value of Gini Index with the following steps:

1. For each non-empty column, form the sum of the squares of the values in the body of the table and divide by the column sum.
2. Add the values obtained for all columns and divided by , the size of the database.
3. Subtract the total from 1.

As an example, with reference to the frequency table as mentioned just.

75

So,

Illustration: Calculating γ using Frequency Table

Thus, the reduction in the value of Gini Index on splitting attribute is

where

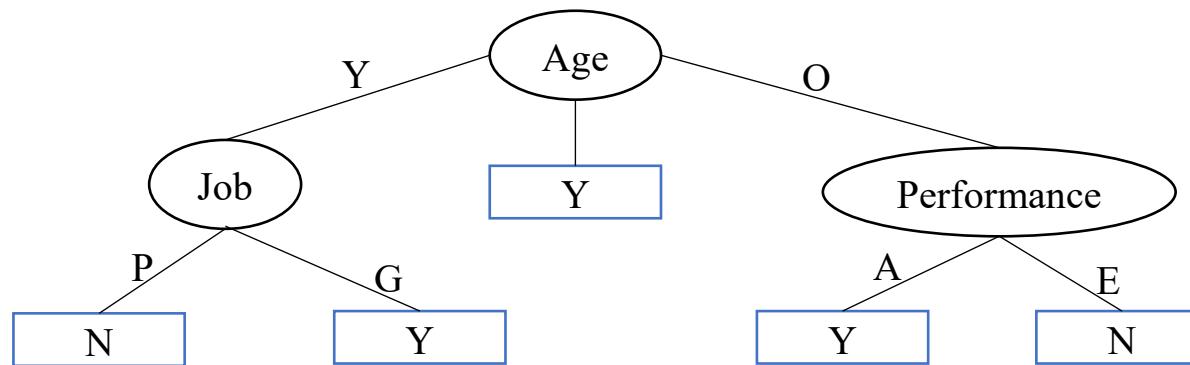
The calculation can be extended to other attributes in the OTPH database and is left as an exercise.



Decision Trees with ID3 and CART Algorithms

Example 20.3 : Comparing Decision Trees of EMP Data set

Compare two decision trees obtained using ID3 and CART for the EMP dataset. The decision tree according to ID3 is given for your ready reference (subject to the verification)



Decision Tree using ID3

?

Decision Tree using CART

Algorithm C4.5

Algorithm C4.5 : Introduction

- J. Ross Quinlan, a researcher in machine learning developed a decision tree induction algorithm in 1984 known as ID3 (Iterative Dichotomizer 3).
- Quinlan later presented C4.5, a successor of ID3, addressing some limitations in ID3.
- ID3 uses information gain measure, which is, in fact **biased towards splitting attribute having a large number of outcomes**.
- For example, if an attribute has distinct values for all tuples, then it would result in a large number of partitions, each one containing just one tuple.
 - In such a case, note that each partition is pure, and hence the purity measure of the partition, that is

Algorithm C4.5 : Introduction

Example 20.4 : Limitation of ID3

In the following, each tuple belongs to a unique class. The splitting on A is shown.

A	-----	class
a_1		
a_2		
⋮		
a_j		
⋮		
a_n		

a_1 ----- |
 a_2 ----- |
⋮
 a_j ----- |
 a_n ----- |

$E(D_j) = l \log_2 l$
 $= 0$

Thus, $E(D_j)$ is maximum in such a situation.
IIITS: Data Analytics

Algorithm: C 4.5 : Introduction

- Although, the previous situation is an extreme case, intuitively, we can infer that **ID3 favours splitting attributes having a large number of values**
 - compared to other attributes, which have a less variations in their values.
- Such a partition appears to be useless for classification.
- This type of problem is called **overfitting problem**.

Note:

Decision Tree Induction Algorithm ID3 may suffer from overfitting problem.

Algorithm: C 4.5 : Introduction

- The overfitting problem in ID3 is due to the measurement of information gain.
- In order to reduce the effect of the use of the bias due to the use of information gain, C4.5 uses a different measure called **Gain Ratio**, denoted as .
- Gain Ratio is a kind of normalization to information gain using a **split information**.

Algorithm: C4.5 : Gain Ratio

Definition 20.3: Gain Ratio

The gain ratio can be defined as follows. We first define **split information** as

Here, m is the number of distinct values in A .

The gain ratio is then defined as where Δ denotes the information gain on splitting the attribute A in the dataset D .

Physical Interpretation of (D)

(D)

- The value of split information depends on
 - the number of (distinct) values an attribute has and
 - how uniformly those values are distributed.
- In other words, it represents the **potential information** generated by splitting a data set D into m partitions, corresponding to the m outcomes of on attribute A .
- Note that for each outcome, it considers the number of tuples having that outcome with respect to the total number of tuples in D .

Physical Interpretation of (D)

Example 20.5 : Split information (D)

- To illustrate (D), let us examine the case where there are 32 instances and splitting an attribute A which has , , and sets of distinct values.
 - Distribution 1 : Highly non-uniform distribution of attribute values

Frequency	32	0	0	0

- Distribution 2

$$(D) = () 1 = 0$$

Frequency	16	16	0	0

$$(D) = ()() = 1$$

Physical Interpretation of (D)

Distribution 3

Frequency	16	8	8	0

$$(D) = ()() 1.5$$

- Distribution 4

Frequency	16	8	4	4

- Distribution 5: Uniform distribution of attribute values

$$(D) = 1.75$$

Frequency	8	8	8	8

$$(D) = ()^*4 = () .0$$

Physical Interpretation of (D)

- In general, if there are m attribute values, each occurring equally frequently, then the split information is .
- Based on the Example 20.5, we can summarize our observation on split information as under:
 - Split information is 0 when there is a single attribute value. It is a trivial case and implies *the minimum possible value of split information*.
 - For a given data set, when instances are uniformly distributed with respect to the attribute values, split information increases as the number of different attribute values increases.
 - The maximum value of split information occur when there are many possible attribute values, all are equally frequent.

Note:

- Split information varies between 0 and $\log_2 m$ (both inclusive)

Physical Interpretation of

- Information gain signifies how much information will be gained on partitioning the values of attribute A
 - Higher information gain means splitting of A is more desirable.
- On the other hand, split information forms the denominator in the gain ratio formula.
 - This implies that higher the value of split information is, lower the gain ratio.
 - In turns, it decreases the information gain.
- Further, information gain is large when there are many distinct attribute values.
 - When many distinct values, split information is also a large value.
 - This way split information reduces the value of gain ratio, thus resulting a balanced value for information gain.
- Like information gain (in ID3), the attribute with the maximum gain ratio is selected as the splitting attribute in C4.5.

Calculation of using Frequency Table

- The frequency table can be used to calculate the gain ratio for a given data set and an attribute.
- We have already learned the calculation of information gain using Frequency Table.
- To calculate gain ratio, in addition to information gain, we are also to calculate split information.
- This split information can be calculated from frequency table as follows.
- For each non-zero column sum say \parallel contribute \parallel for the j -th column (i.e., the j -th value of the attribute). Thus the split information is

$$(D) =$$

If there are m -columns in the frequency table.

Practice:

Using Gain ratio as the measurement of splitting attributes, draw the decision trees for OPTH and EMP data sets. Give calculation of gain ratio at each node.

Summary of Decision Tree Induction Algorithms

- We have learned the building of a decision tree given a training data.
 - The decision tree is then used to classify a test data.
- For a given training data D , the important task is to build the decision tree so that:
 - All test data can be classified accurately
 - The tree is balanced and with as minimum depth as possible, thus the classification can be done at a faster rate.
- In order to build a decision tree, several algorithms have been proposed. These algorithms differ from the chosen splitting criteria, so that they satisfy the above mentioned objectives as well as the decision tree can be induced with minimum time complexity. We have studied three decision tree induction algorithms namely ID3, CART and C4.5. A summary of these three algorithms is presented in the following table.

Table 20.6

Algorithm	Splitting Criteria	Remark
ID3	<p>Information Gain</p> $I(D)$ Where $=$ Entropy of D (a measure of uncertainty) = where D is with set of k classes , , ... , and $=$; Here, τ is the set of tuples with class in D . $I(D) =$ Weighted average entropy when D is partitioned on the values of attribute $A = \{ \dots \}$ Here, m denotes the distinct values of attribute A .	<ul style="list-style-type: none"> The algorithm calculates $I(D)$ for all $a \in D$ and choose that attribute which has maximum $I(D)$. The algorithm can handle both categorical and numerical attributes. It favors splitting those attributes, which has a large number of distinct values.

Algorithm	Splitting Criteria	Remark
CART	<p>Gini Index (D) where = Gini index (a measure of impurity) =</p> <p>Here, = and D is with k number of classes $G_A(D) =) +),$ when D is partitioned into two data sets and based on some values of attribute A.</p>	<ul style="list-style-type: none"> The algorithm calculates all binary partitions for all possible values of attribute A and choose that binary partition which has the maximum The algorithm is computationally very expensive when the attribute A has a large number of values.

Algorithm	Splitting Criteria	Remark
C4.5	<p>Gain Ratio</p> <p>where</p> <ul style="list-style-type: none"> = Information gain of D (same as in ID3, and = splitting information = when D is partitioned into , , ... , partitions corresponding to m distinct attribute values of A. 	<ul style="list-style-type: none"> • The attribute A with maximum value of is selected for splitting. • Splitting information is a kind of normalization, so that it can check the biasness of information gain towards the choosing attributes with a large number of distinct values.

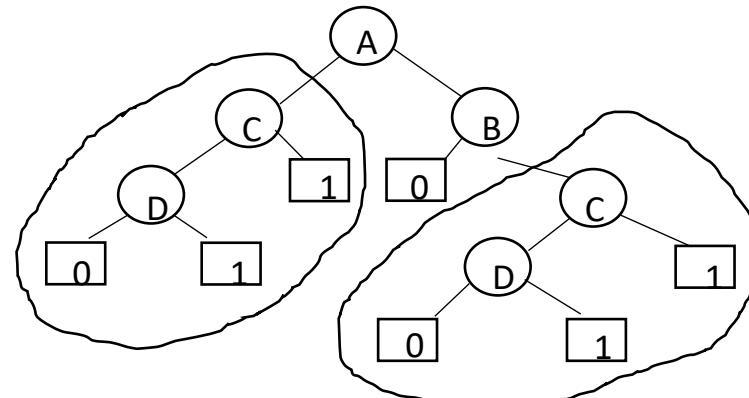
In addition to this, we also highlight few important characteristics of decision tree induction algorithms in the following.

Notes on Decision Tree Induction algorithms

- 1. Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms **employ a heuristic based approach** to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.
- 2. Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.
- 3. Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to be chosen for splitting.
- 4. Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of $O(d)$, where d is the maximum depth of the tree.

Notes on Decision Tree Induction algorithms

5. **Data Fragmentation Problem:** Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.
6. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



Reference

- The detail material related to this lecture can be found in

Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2015.

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison-Wesley, 2014

Any question?

Feature Translation

More Regression and Classification approaches

Linear Regression

- Given data with n dimensional variables and 1 target-variable (real number)

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$$

Where $\mathbf{x} \in \Re^n, y \in \Re$

- The objective: Find a function f that returns the best fit. $f: \Re^n \rightarrow \Re$
- Assume that the relationship between X and y is approximately linear. The model can be represented as (w represents coefficients and b is an intercept)

$$f(w_1, \dots, w_n, b) = y = \mathbf{w} \cdot \mathbf{x} + b + \varepsilon$$

Linear Regression

- To find the best fit, we minimize the sum of squared errors → Least square estimation

$$\min \sum_{i=1}^m (y_i - \hat{y}_i)^2 = \sum_{i=1}^m (y_i - (\mathbf{w} \cdot \mathbf{x}_i + b))^2$$

- The solution can be found by solving
 $\hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$
(By taking the derivative of the above objective function w.r.t. \mathbf{w})
- In MATLAB, the back-slash operator computes a least square solution.

Linear Regression

$$\min \sum_{i=1}^m (y_i - \hat{y}_i)^2 = \sum_{i=1}^m (y_i - (\hat{\mathbf{w}} \cdot \mathbf{x}_i + \hat{b}))^2$$



- To avoid over-fitting, a regularization term can be introduced (minimize a magnitude of w)

- LASSO: $\min \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^n |w_j|$

- Ridge regression: $\min \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^n |\mathbf{w}_j^2|$

Support Vector Regression

- Find a function, $f(x)$, with at most ε -deviation from the target y

The problem can be written as a convex optimization problem

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

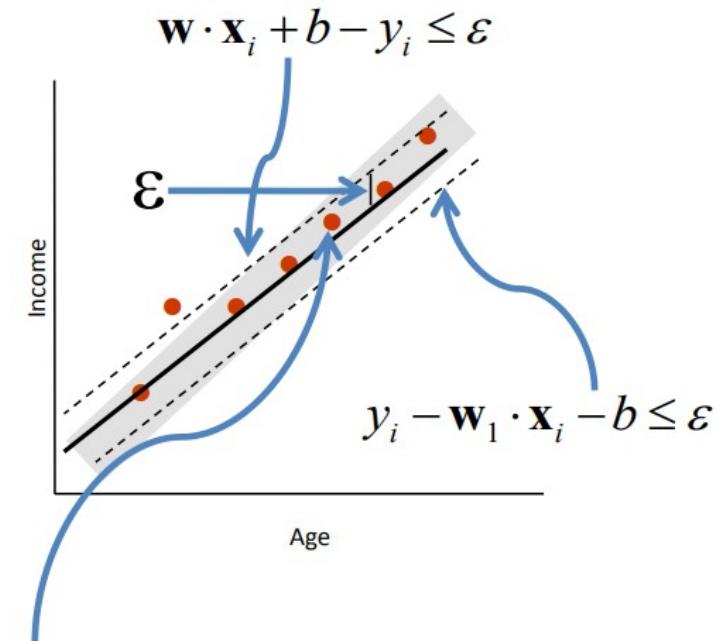
$$s.t. y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \leq \varepsilon;$$

$$\mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \leq \varepsilon;$$

C: trade off the complexity

What if the problem is not feasible?

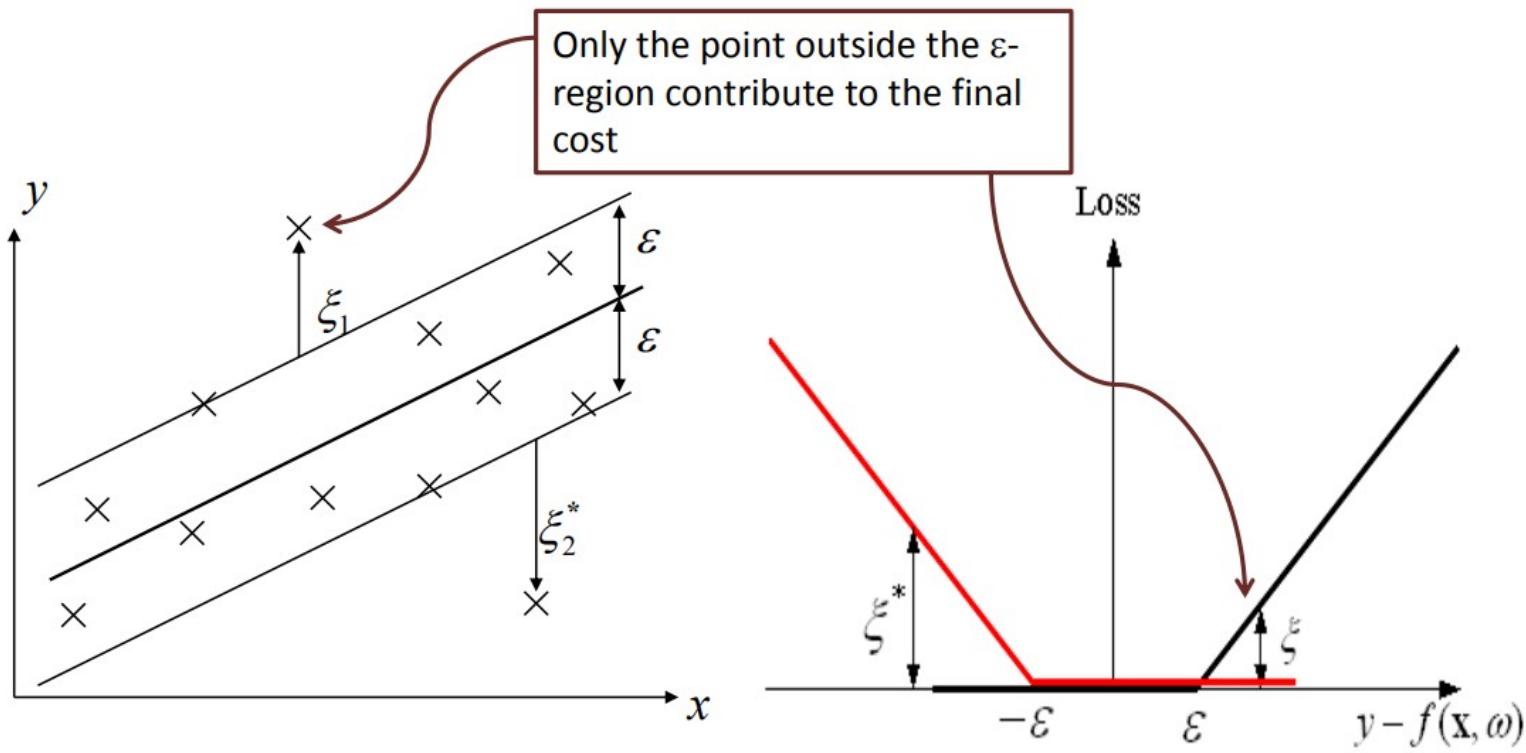
We can introduce slack variables
(similar to soft margin loss function).



We do not care about errors as long as they are less than ε

Support Vector Regression

Assume linear parameterization $f(\mathbf{x}, \omega) = \mathbf{w} \cdot \mathbf{x} + b$



$$L_\epsilon(y, f(\mathbf{x}, \omega)) = \max(|y - f(\mathbf{x}, \omega)| - \epsilon, 0)$$

Soft margin

Given training data

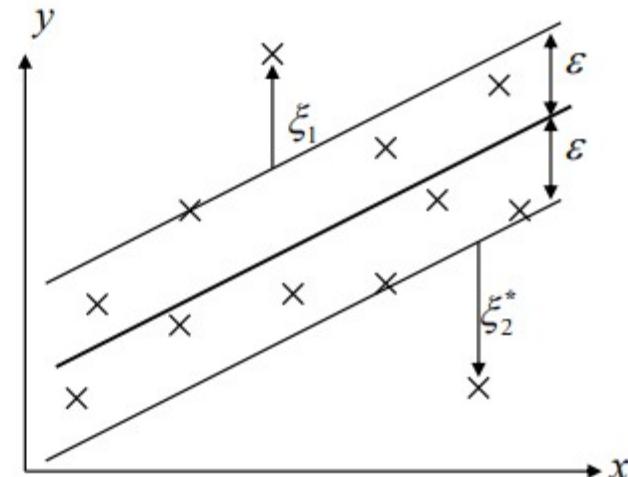
$$(\mathbf{x}_i, y_i) \quad i = 1, \dots, m$$

Minimize

$$\frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*)$$

Under constraints

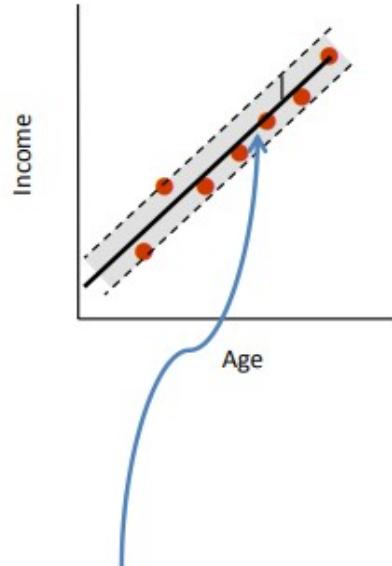
$$\begin{cases} y_i - (\mathbf{w} \cdot \mathbf{x}_i) - b \leq \varepsilon + \xi_i \\ (\mathbf{w} \cdot \mathbf{x}_i) + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, i = 1, \dots, m \end{cases}$$



Linear versus Non-linear SVR

- **Linear case**

$$f : \text{age} \rightarrow \text{income}$$

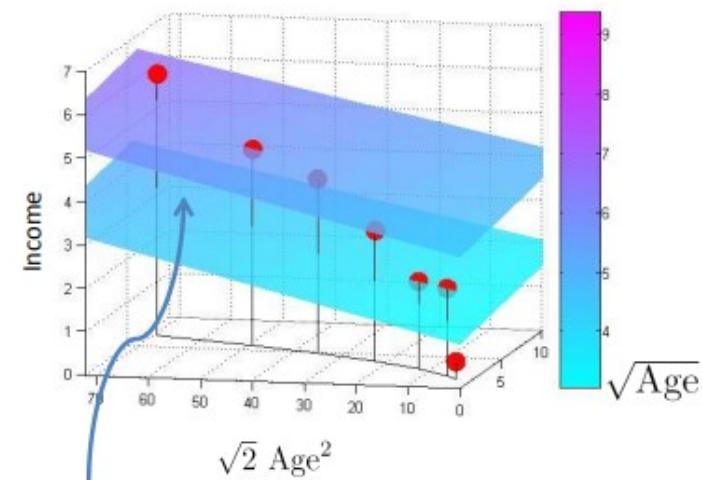


$$y_i = \mathbf{w}_1 \cdot \mathbf{x}_i + b$$

- **Non-linear case**

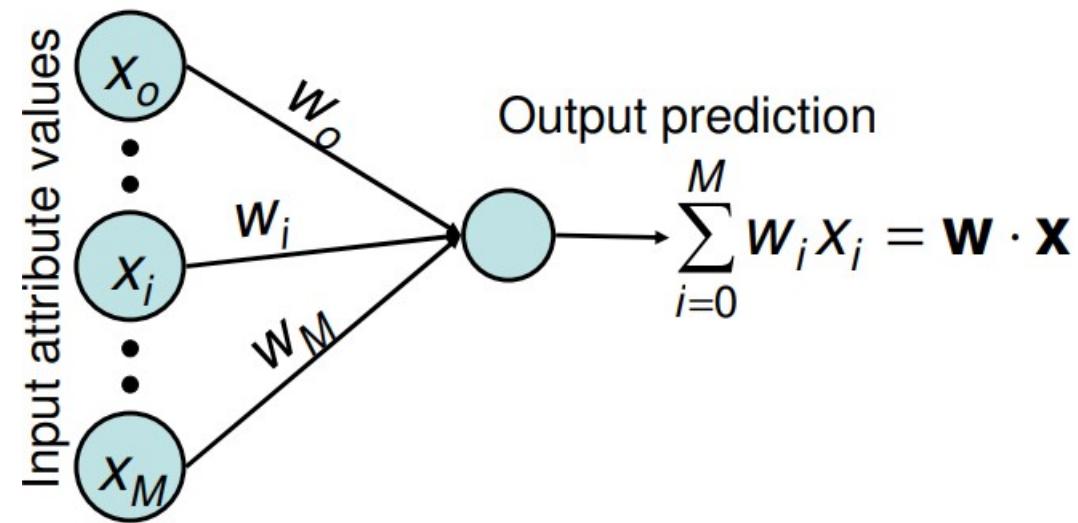
- Map data into a higher dimensional space, e.g.,

$$f : (\sqrt{\text{age}}, \sqrt{2}\text{age}^2) \rightarrow \text{income}$$



$$y_i = \mathbf{w}_1 \sqrt{\mathbf{x}_i} + \mathbf{w}_2 \sqrt{2\mathbf{x}_i^2} + b$$

Regression: Neural Networks



Regression: Neural Networks

Perceptron Training

- Given input training data x^k with corresponding value y^k
- 1. Compute error:

$$\delta_k \leftarrow y^k - \mathbf{w} \cdot \mathbf{x}^k$$

- 2. Update NN weights:

$$w_i \leftarrow w_i + \alpha \delta_k x_i^k$$

Regression: Neural Networks

- Update has many names: delta rule, gradient rule, LMS rule
- Update is guaranteed to converge to the best linear fit (global minimum of E)
- Of course, there are more direct ways of solving the linear regression problem by using linear algebra techniques. It boils down to a simple matrix inversion (not shown here).
- In fact, the perceptron training algorithm can be much, much slower than the direct solution
- Use of MLP with Backpropagation for Non-Linear Regression

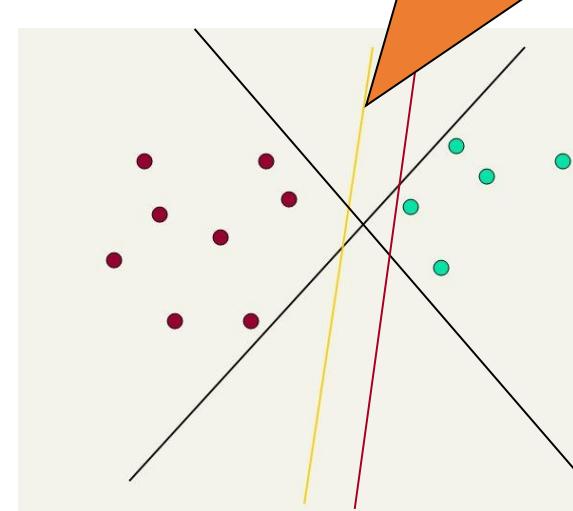
Perceptron Learning Algorithm

- First neural network learning model in the 1960's
- Simple and limited (single layer model)
- Basic concepts are similar for multi-layer models, so this is a good learning tool
- Still used in some current applications (large business problems, where intelligibility is needed, etc.)

Linear classifiers: Which Hyperplane?

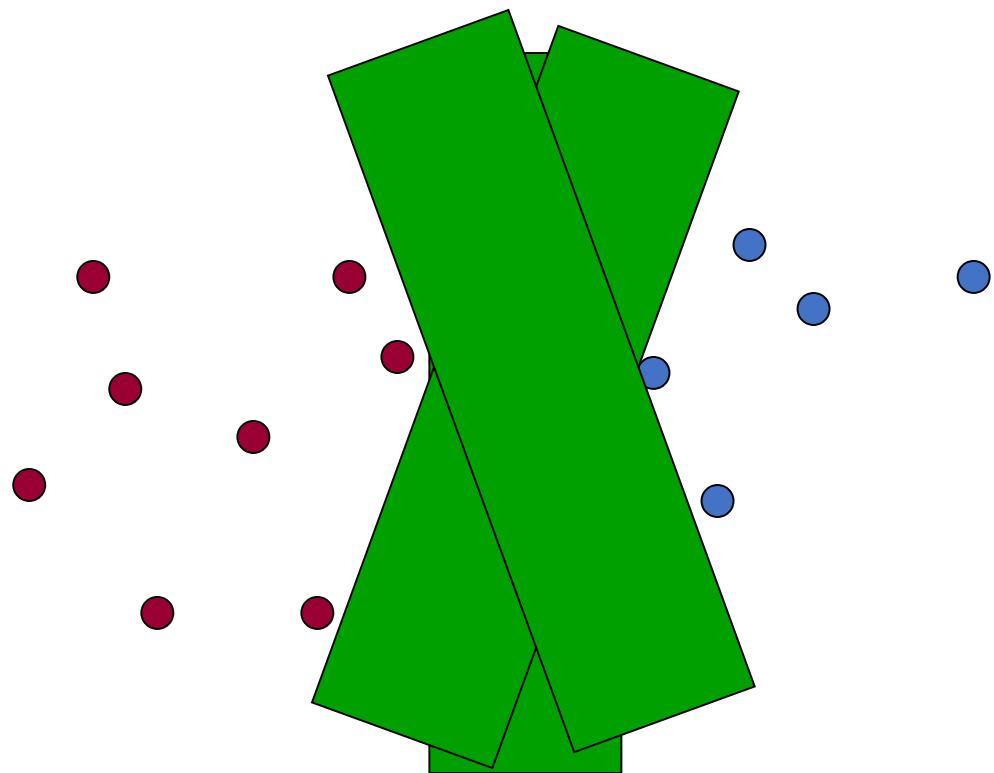
- Lots of possible solutions for a, b, c .
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
 - E.g., perceptron
- Support Vector Machine (SVM) finds an optimal* solution.
 - Maximizes the distance between the hyperplane and the “difficult points” close to decision boundary
 - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

This line represents the decision boundary:
 $ax + by - c = 0$



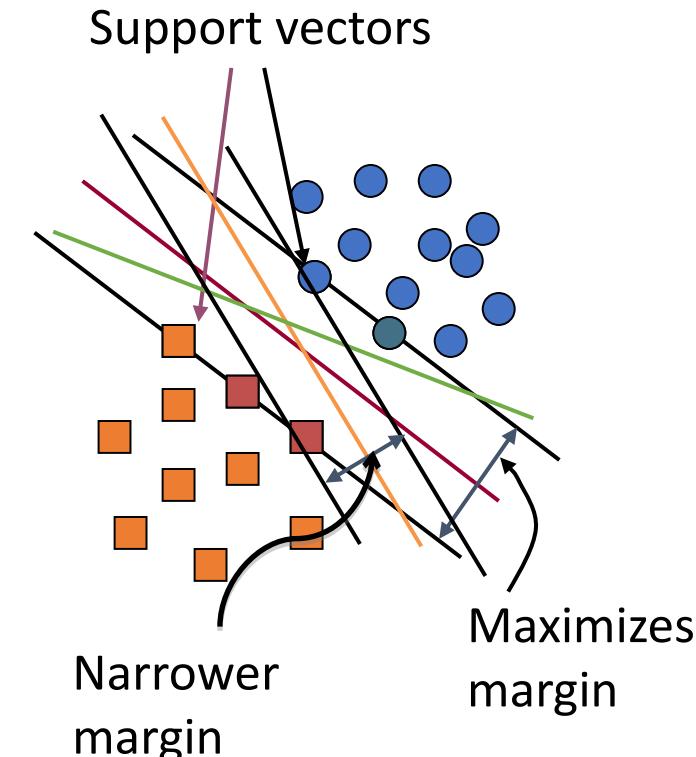
Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
 - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*



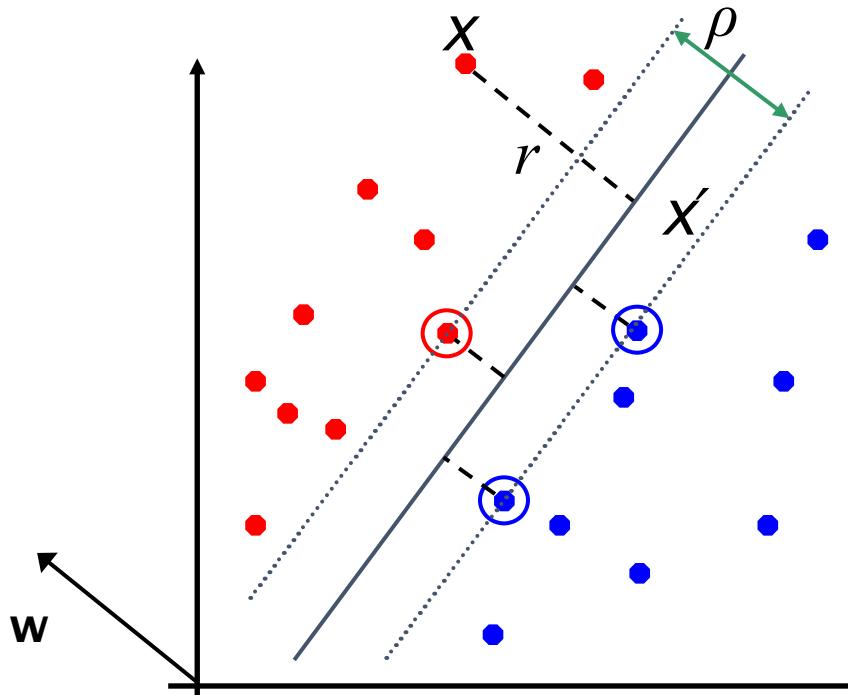
*but other discriminative methods often perform very similarly

Maximum Margin: Formalization

- \mathbf{w} : decision hyperplane normal vector
- \mathbf{x}_i : data point i
- y_i : class of data point i (+1 or -1) NB: Not 1/0
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)$
- Functional margin of \mathbf{x}_i is: $y_i (\mathbf{w}^T \mathbf{x}_i + b)$
 - But note that we can increase this margin simply by scaling \mathbf{w}, b
- Functional margin of dataset is twice the minimum functional margin for any point
 - The factor of 2 comes from measuring the whole width of the margin

Geometric Margin

- Distance from example to the separator is $r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are **support vectors**.
- **Margin ρ** of the separator is the width of separation between support vectors of classes.



Derivation of finding r :

Dotted line $\mathbf{x}' - \mathbf{x}$ is perpendicular to decision boundary so parallel to \mathbf{w} .
Unit vector is $\mathbf{w}/|\mathbf{w}|$, so line is $r\mathbf{w}/|\mathbf{w}|$.

$$\mathbf{x}' = \mathbf{x} - yr\mathbf{w}/|\mathbf{w}|.$$

$$\mathbf{x}' \text{ satisfies } \mathbf{w}^T \mathbf{x}' + b = 0.$$

$$\text{So } \mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0 \\ \text{Recall that } |\mathbf{w}| = \sqrt{\mathbf{w}^T \mathbf{w}}.$$

$$\text{So } \mathbf{w}^T \mathbf{x} - yr\sqrt{\mathbf{w}^T \mathbf{w}} + b = 0$$

So, solving for r gives:

$$r = y(\mathbf{w}^T \mathbf{x} + b)/|\mathbf{w}|$$

Linear SVM Mathematically

The linearly separable case

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x}_i, y_i)\}$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

$$r = y \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Linear Support Vector Machine (SVM)

- **Hyperplane**

$$\mathbf{w}^T \mathbf{x} + b = 0$$

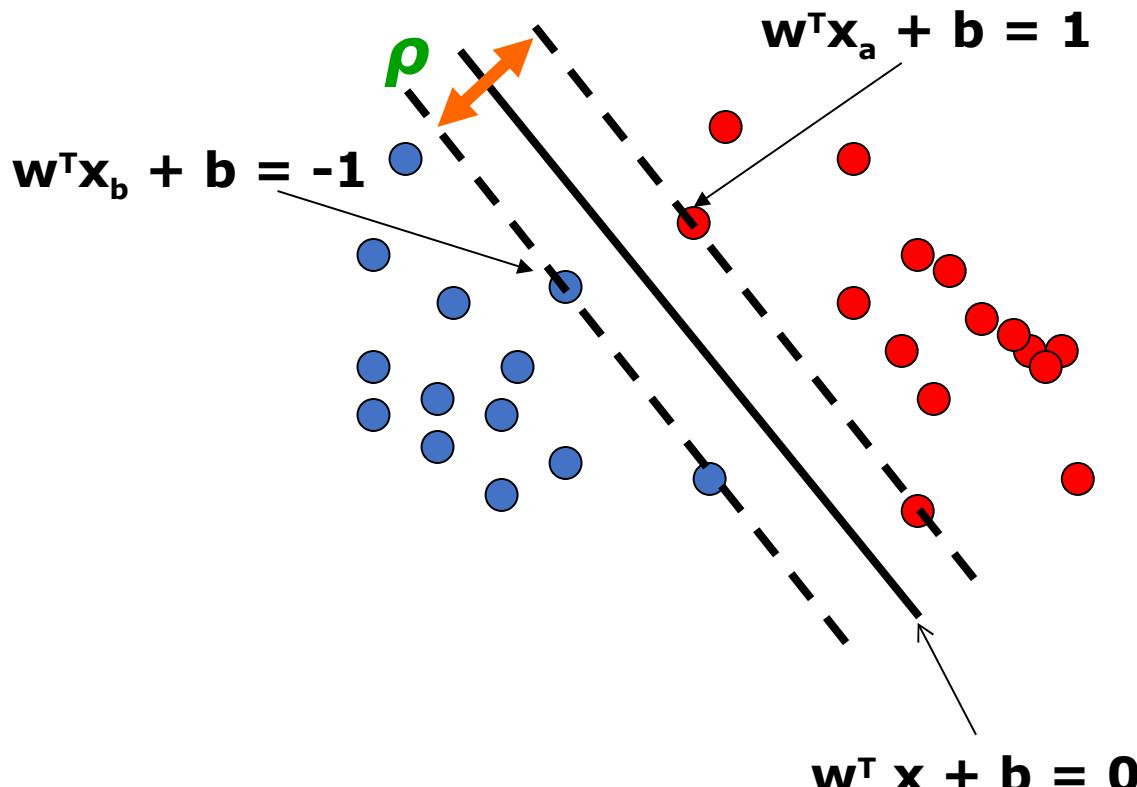
- **Extra scale constraint:**

$$\min_{i=1,\dots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$$

- This implies:

$$\mathbf{w}^T (\mathbf{x}_a - \mathbf{x}_b) = 2$$

$$\rho = \| \mathbf{x}_a - \mathbf{x}_b \|_2 = 2 / \| \mathbf{w} \|_2$$



Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem*:

Find \mathbf{w} and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \quad \text{is maximized; and for all } \{(\mathbf{x}_i, y_i)\}$$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1; \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

- A better formulation ($\min ||\mathbf{w}|| = \max 1/ ||\mathbf{w}||$):

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \quad \text{is minimized;}$$

$$\text{and for all } \{(\mathbf{x}_i, y_i)\}: \quad y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Solving the Optimization Problem

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \dots \alpha_N$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

$$(1) \quad \sum \alpha_i y_i = 0$$

$$(2) \quad \alpha_i \geq 0 \text{ for all } \alpha_i$$

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \quad b = y_k - \mathbf{w}^T \mathbf{x}_k \text{ for any } \mathbf{x}_k \text{ such that } \alpha_k \neq 0$$

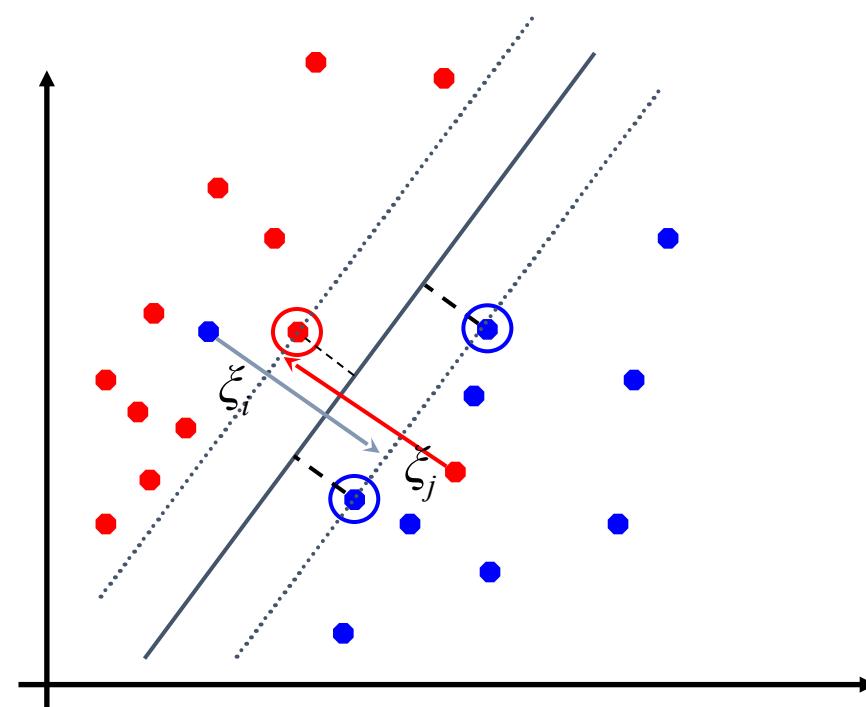
- Each non-zero α_i indicates that corresponding \mathbf{x}_i is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point \mathbf{x} and the support vectors \mathbf{x}_i
 - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^T \mathbf{x}_j$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
 - Let some points be moved to where they belong, at a cost
 - Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- The new formulation incorporating slack variables:

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \text{ for all } i$$

- Parameter C can be viewed as a way to control overfitting
 - A regularization term

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

$$(1) \quad \sum \alpha_i y_i = 0$$

$$(2) \quad 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, \mathbf{x}_i with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$$b = y_k(1 - \xi_k) - \mathbf{w}^T \mathbf{x}_k \text{ where } k = \operatorname{argmax}_{k'} \alpha_{k'}$$

\mathbf{w} is not needed explicitly for classification!

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

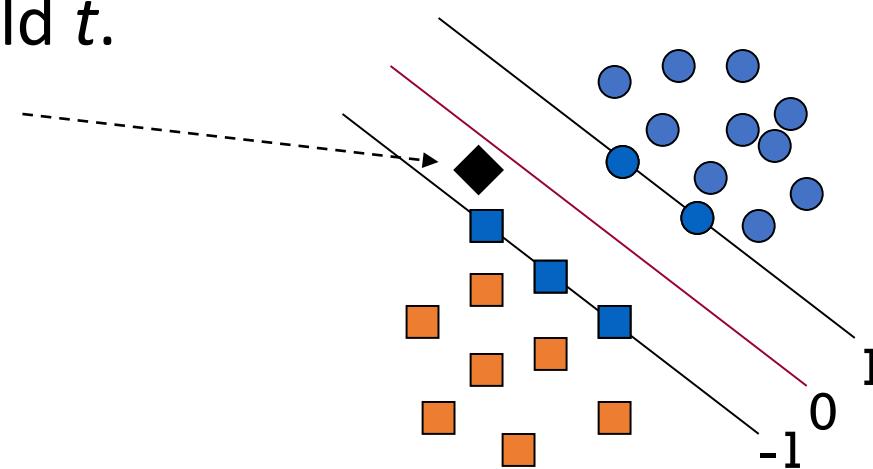
Classification with SVMs

- Given a new point \mathbf{x} , we can score its projection onto the hyperplane normal:
 - I.e., compute score: $\mathbf{w}^\top \mathbf{x} + b = \sum \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b$
 - Decide class based on whether < 0 or > 0
- Can set confidence threshold t .

Score $> t$: yes

Score $< -t$: no

Else: don't know



Linear SVMs: Summary

- The classifier is a *separating hyperplane*.
- The most “important” training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points \mathbf{x}_i are support vectors with non-zero Lagrangian multipliers α_i ,
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1, \dots, \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

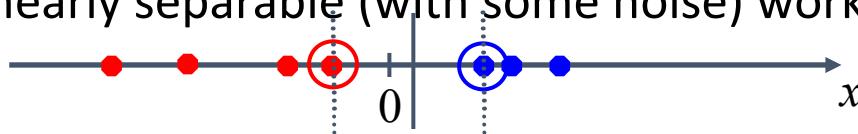
$$(1) \quad \sum \alpha_i y_i = 0$$

$$(2) \quad 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Non-linear SVMs

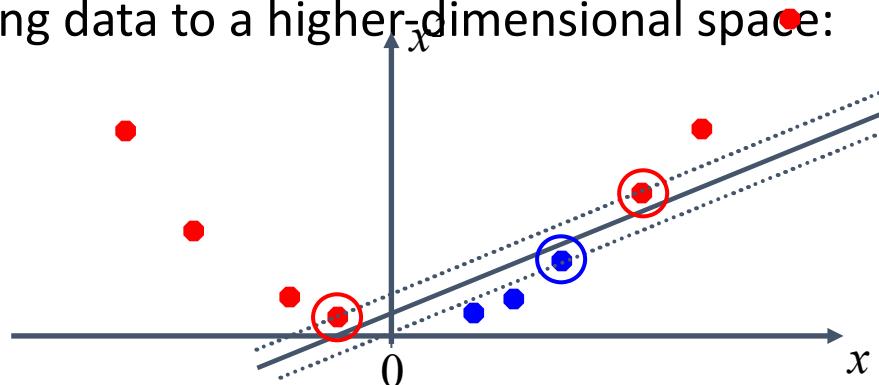
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?

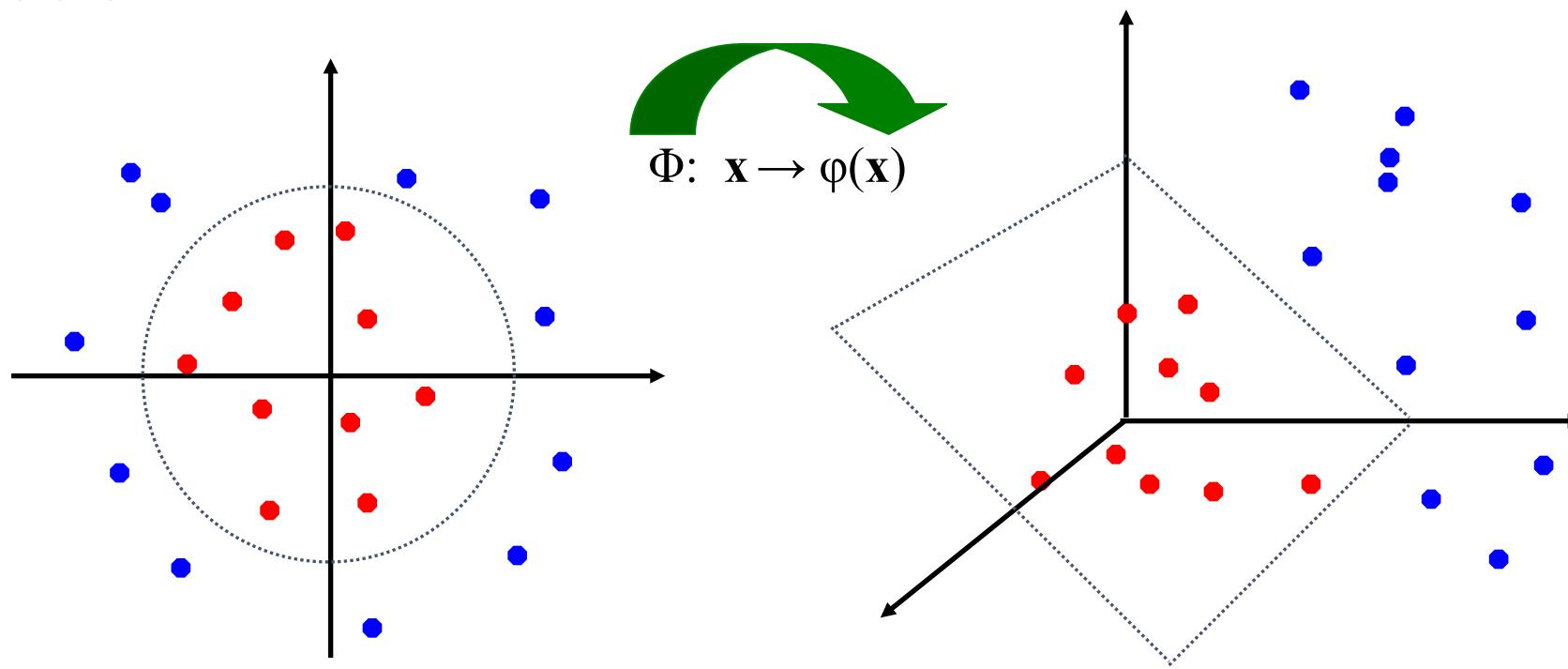


- How about ... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The “Kernel Trick”

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

2-dimensional vectors $\mathbf{x} = [x_1 \ x_2]$; let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^2 = 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} =$$

$$= [1 \ x_{i1}^2 \ \sqrt{2} x_{i1} x_{i2} \ x_{i2}^2 \ \sqrt{2} x_{i1} \ \sqrt{2} x_{i2}]^\top [1 \ x_{j1}^2 \ \sqrt{2} x_{j1} x_{j2} \ x_{j2}^2 \ \sqrt{2} x_{j1} \ \sqrt{2} x_{j2}]$$

$$= \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \quad \text{where } \phi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2} x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2]$$

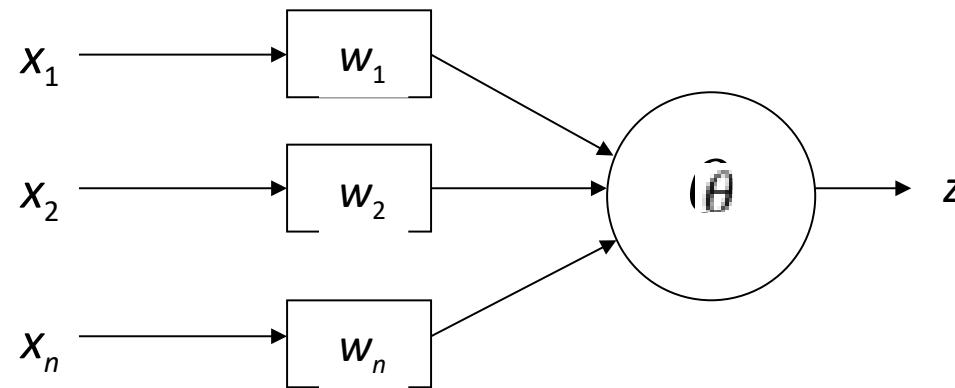
Kernels

- Why use kernels?
 - Make non-separable problem separable.
 - Map data into better representational space
- Common kernels
 - Linear
 - Polynomial $K(x,z) = (1+x^T z)^d$
 - Gives feature conjunctions
 - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

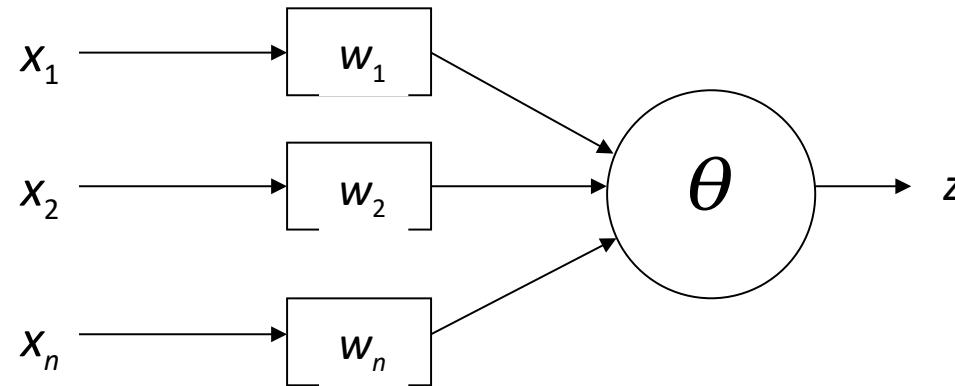
- Haven't been very useful in text classification

Classification: Perceptron Node – Threshold Logic Unit



$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^n x_i w_i < \theta \end{cases}$$

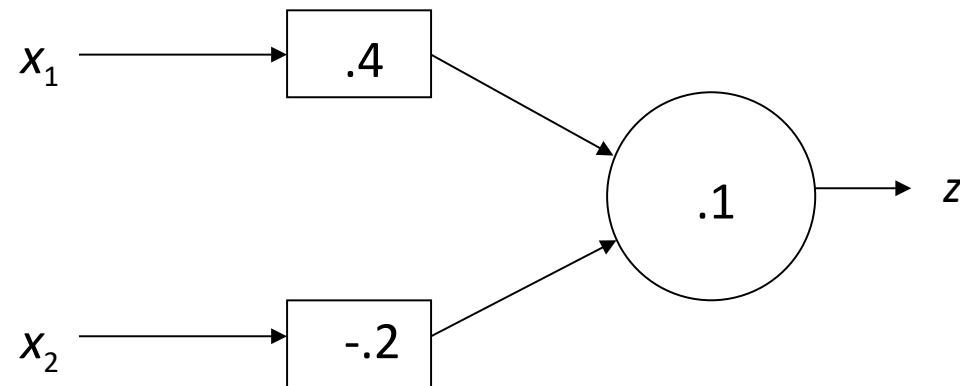
Perceptron Node – Threshold Logic Unit



- Learn weights such that an objective function is maximized.
- What objective function should we use?
- What learning algorithm should we use?

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^n x_i w_i < \theta \end{cases}$$

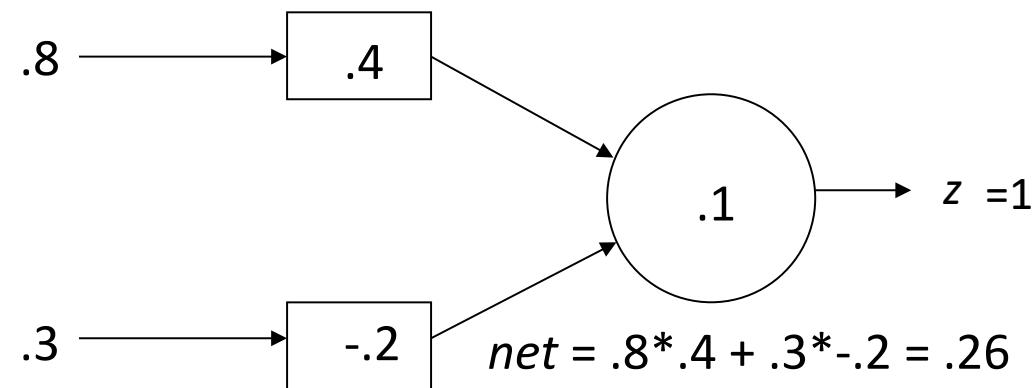
Perceptron Learning Algorithm



x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^n x_i w_i < \theta \end{cases}$$

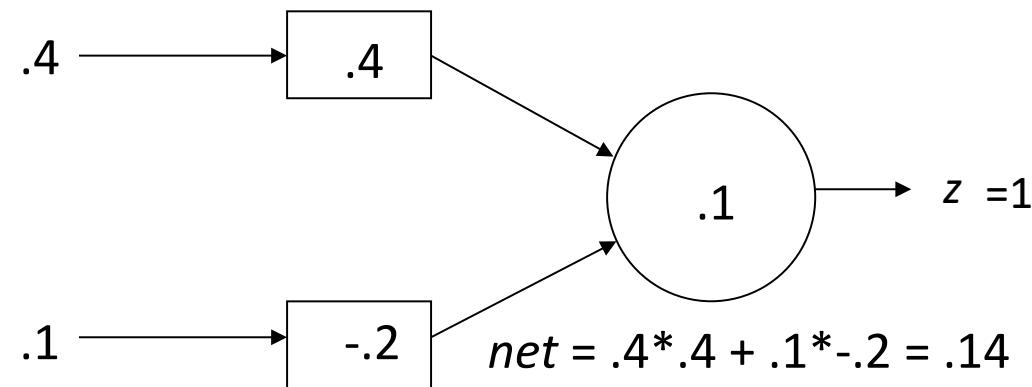
First Training Instance



x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^n x_i w_i < \theta \end{cases}$$

Second Training Instance



x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^n x_i w_i < \theta \end{cases}$$
$$\Delta w_i = (t - z) * c * x_i$$

Perceptron Rule Learning

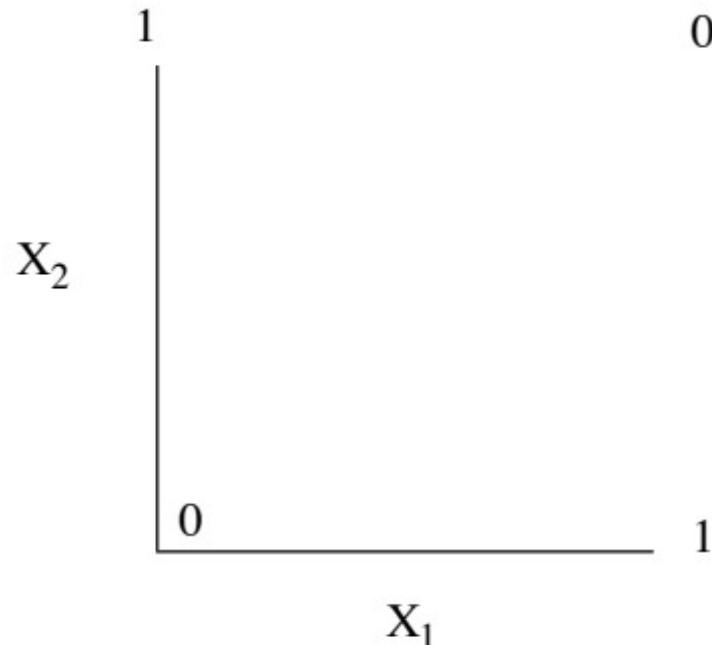
$$\Delta w_i = c(t - z) x_i$$

- Where w_i is the weight from input i to perceptron node, c is the learning rate, t is the target for the current instance, z is the current output, and x_i is i^{th} input
- Least perturbation principle
 - Only change weights if there is an error
 - small c rather than changing weights sufficient to make current pattern correct
 - Scale by x_i
- Create a perceptron node with n inputs
- Iteratively apply a pattern from the training set and apply the perceptron rule
- Each iteration through the training set is an *epoch*
- Continue training until total training set error ceases to improve
- Perceptron Convergence Theorem: Guaranteed to find a solution in finite time if a solution exists

How to Handle Multi-Class Output

- This is an issue with any learning model which only supports binary classification (perceptron, SVM, etc.)
- Create 1 perceptron for each output class, where the training set considers all other classes to be negative examples (**one vs the rest**)
 - Run all perceptrons on novel data and set the output to the class of the perceptron which outputs high
 - If there is a tie, choose the perceptron with the highest net value
- Create 1 perceptron for each pair of output classes, where the training set only contains examples from the 2 classes (**one vs one**)
 - Run all perceptrons on novel data and set the output to be the class with the most wins (votes) from the perceptrons
 - In case of a tie, use the net values to decide
 - Number of models grows by the square of the output classes

Exclusive Or



Is there a dividing hyperplane?

Neural Nets

Brain Computer Interaction- S2022

Acknowledgements: [\(berkeley.edu\)](https://berkeley.edu)

Feature Vectors

x

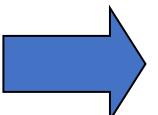
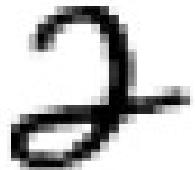
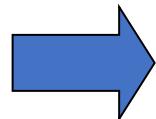
```
Hello,  
  
Do you want free printr  
cartridges? Why pay more  
when you can get them  
ABSOLUTELY FREE! Just
```

$f(x)$

$$\begin{Bmatrix} \# \text{ free} & : 2 \\ \text{YOUR_NAME} & : 0 \\ \text{MISSPELLED} & : 2 \\ \text{FROM_FRIEND} & : 0 \\ \dots \end{Bmatrix}$$

y

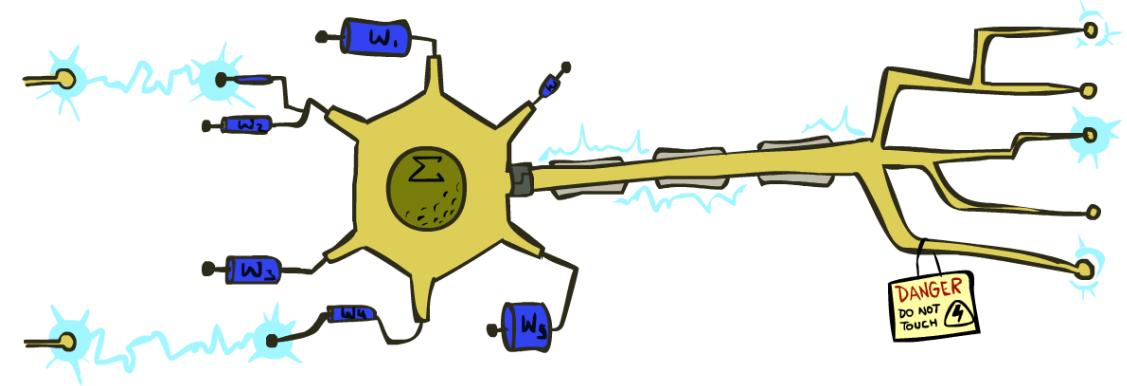
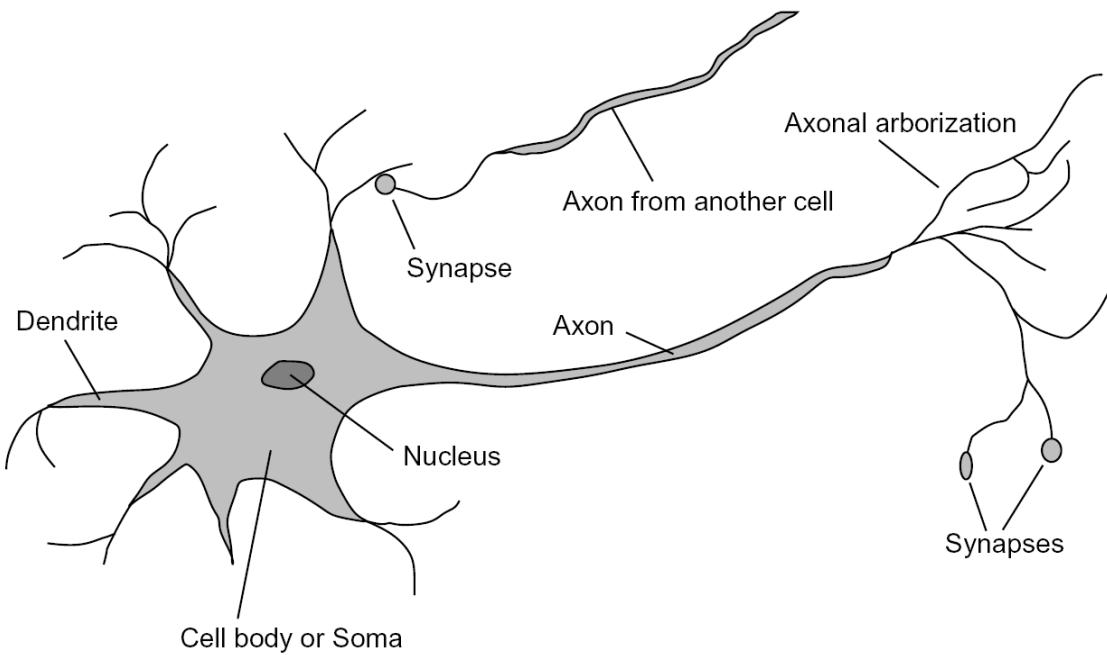
SPAM
or
+


$$\begin{Bmatrix} \text{PIXEL-7,12} & : 1 \\ \text{PIXEL-7,13} & : 0 \\ \dots \\ \text{NUM_LOOPS} & : 1 \\ \dots \end{Bmatrix}$$


"2"

Some (Simplified) Biology

- Very loose inspiration: human neurons

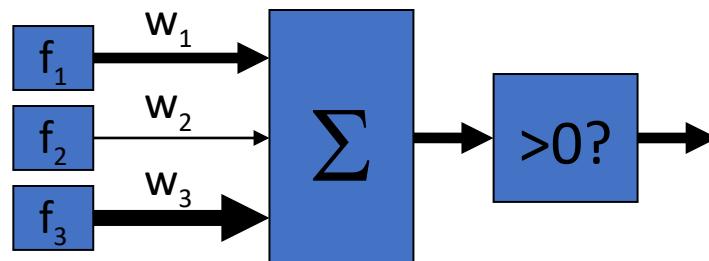


Linear Classifiers

- Inputs are **feature values**
- Each feature has a **weight**
- Sum is the **activation**

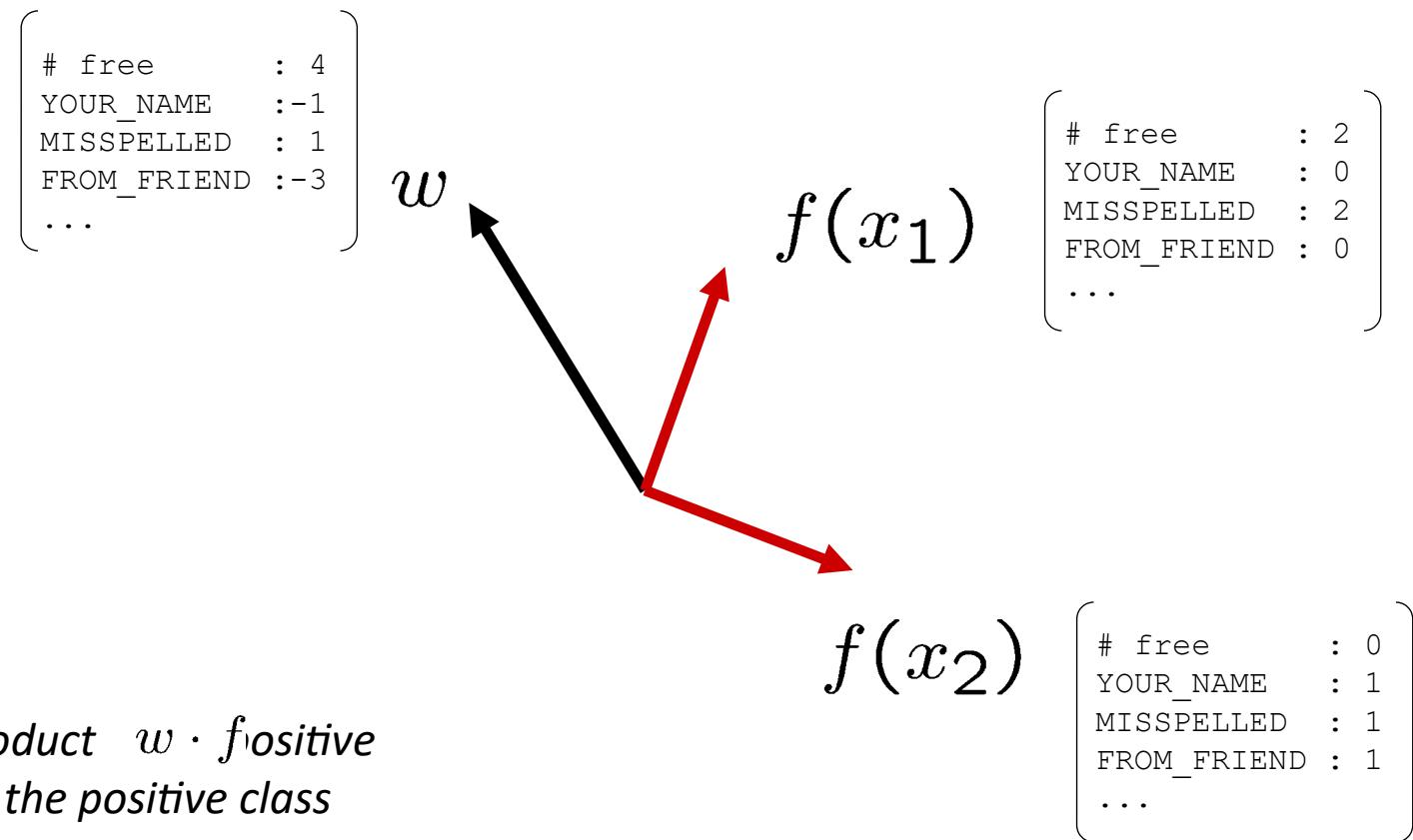
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- If the activation is:
 - Positive, output +1
 - Negative, output -1



Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples



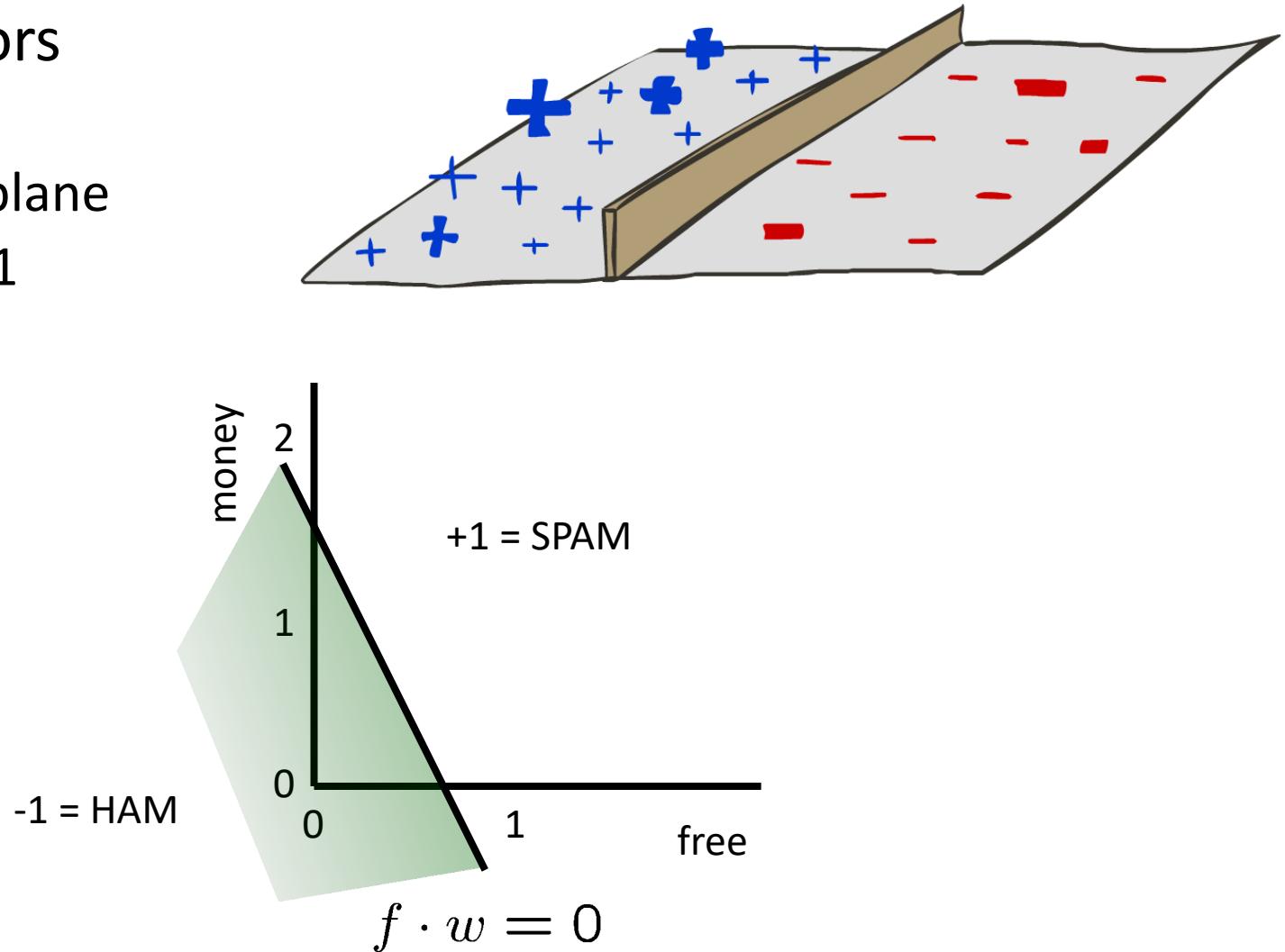
*Dot product $w \cdot f$ positive
means the positive class*

Binary Decision Rule

- In the space of feature vectors
 - Examples are points
 - Any weight vector is a hyperplane
 - One side corresponds to $Y=+1$
 - Other corresponds to $Y=-1$

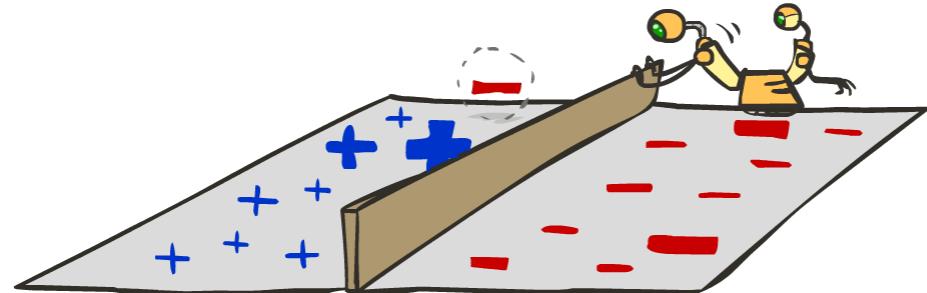
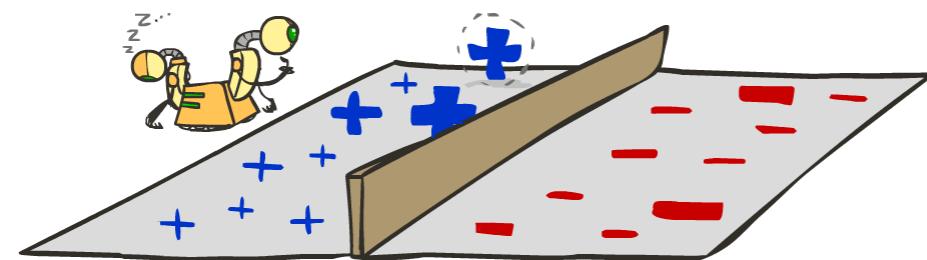
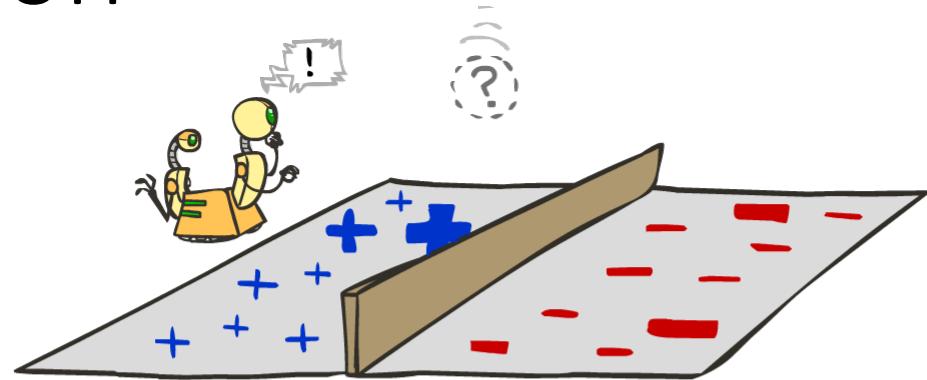
w

BIAS	:	-3
free	:	4
money	:	2
...		



Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
 - Classify with current weights
- If correct (i.e., $y=y^*$), no change!
- If wrong: adjust the weight vector



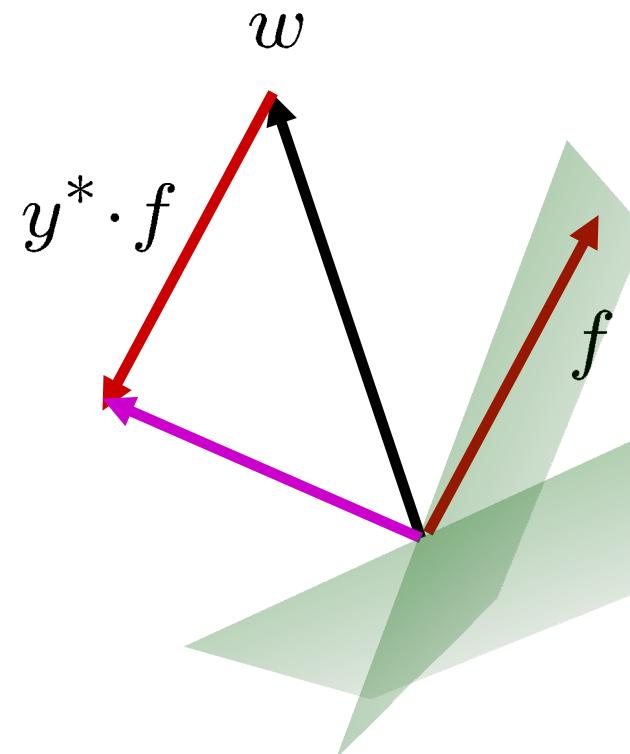
Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
 - Classify with current weights

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

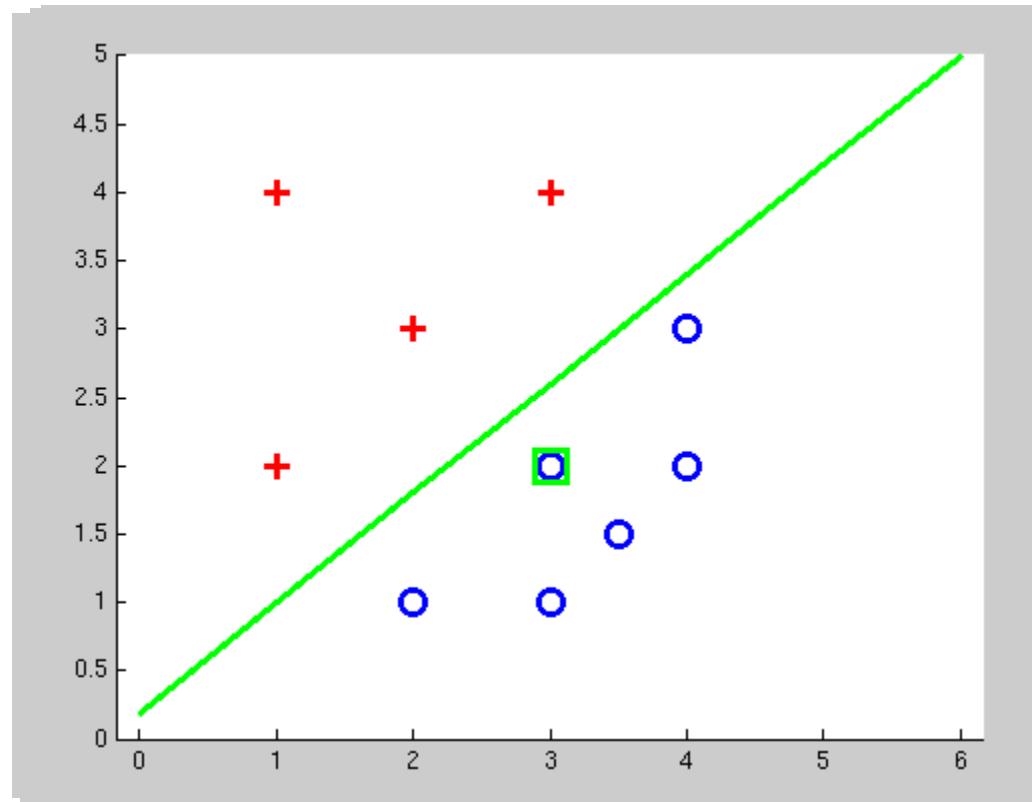
- If correct (i.e., $y=y^*$), no change!
- If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y^* is -1.

$$w = w + y^* \cdot f$$



Examples: Perceptron

- Separable Case



Multiclass Decision Rule

- If we have multiple classes:
 - A weight vector for each class:

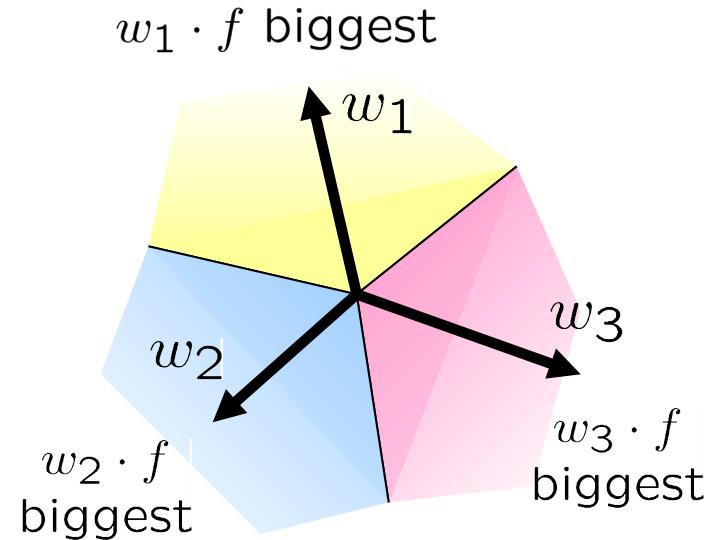
$$w_y$$

- Score (activation) of a class y :

$$w_y \cdot f(x)$$

- Prediction highest score wins

$$y = \arg \max_y w_y \cdot f(x)$$



Binary = multiclass where the negative class has weight zero

Learning: Multiclass Perceptron

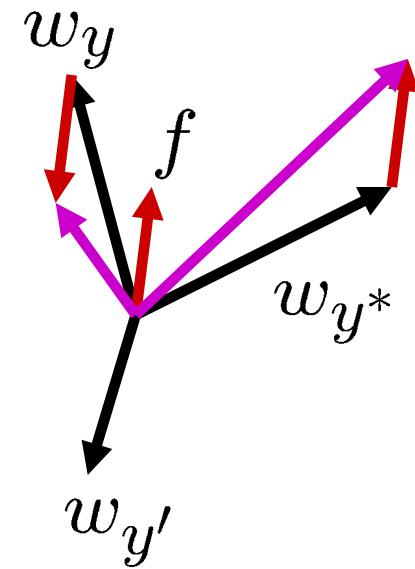
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

$$y = \arg \max_y w_y \cdot f(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer,
raise score of right answer

$$w_y = w_y - f(x)$$

$$w_{y^*} = w_{y^*} + f(x)$$

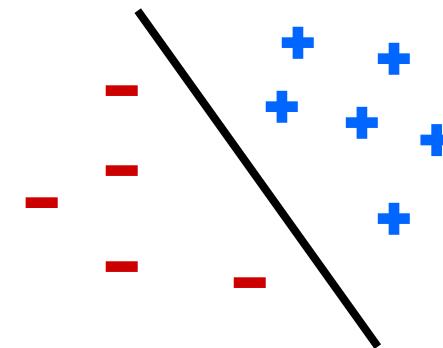


Properties of Perceptrons

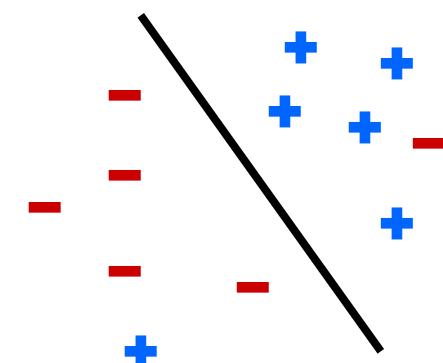
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

Separable

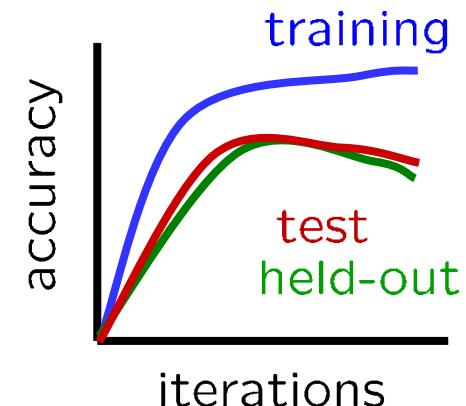
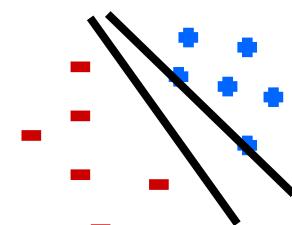
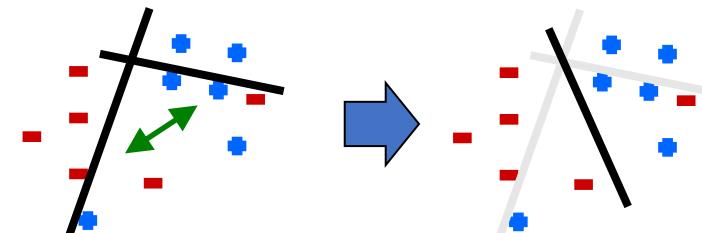


Non-Separable



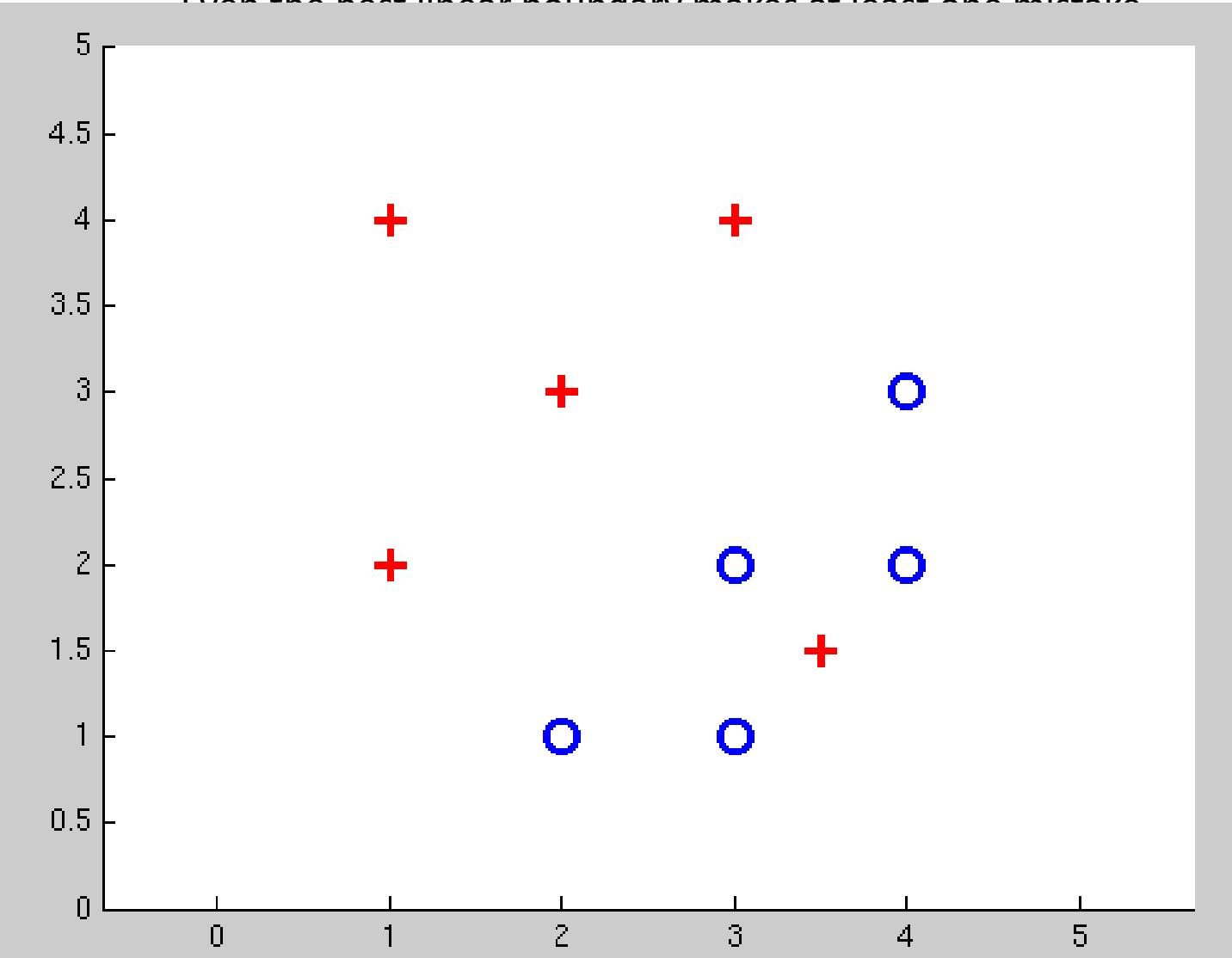
Problems with the Perceptron

- Noise: if the data isn't separable, weights might thrash
 - Averaging weight vectors over time can help (averaged perceptron)
- Mediocre generalization: finds a “barely” separating solution
- Overtraining: test / held-out accuracy usually rises, then falls
 - Overtraining is a kind of overfitting

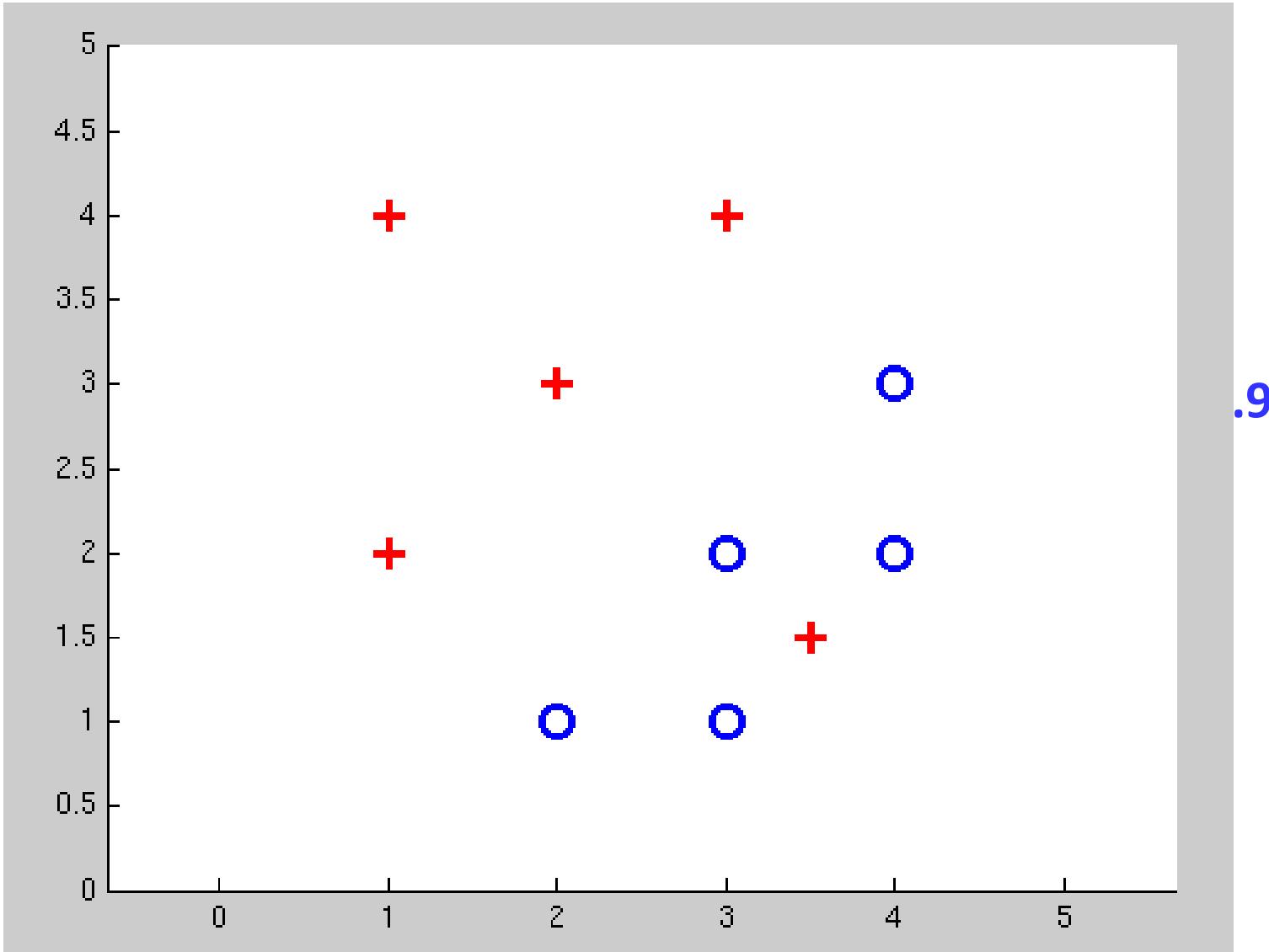


Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake.



Non-Separable Case: Probabilistic Decision

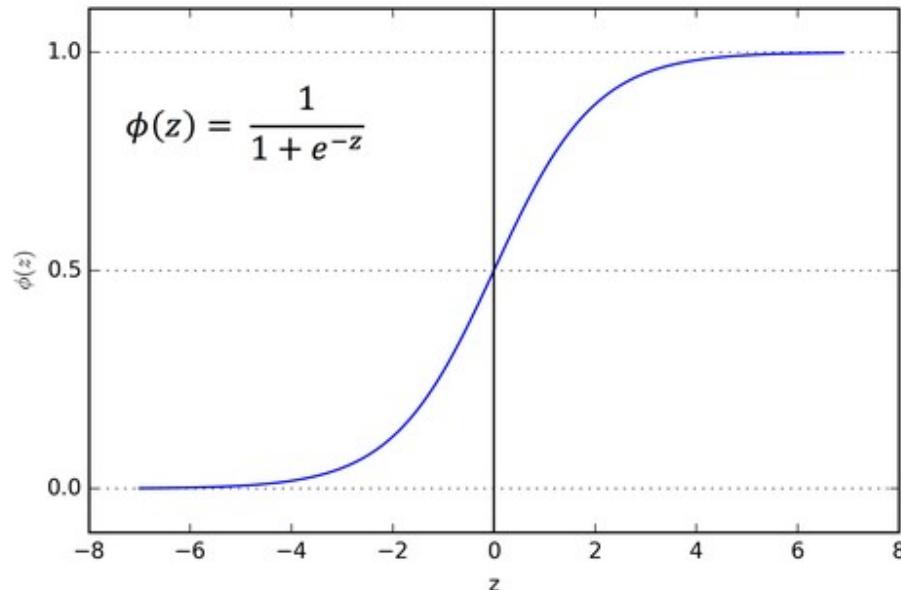


How to get probabilistic decisions?

$$z = w \cdot f(x)$$

- Perceptron scoring:
- If $z = w \cdot f(x)$ very positive \rightarrow want probability going to 1
- If $z = w \cdot f(x)$ very negative \rightarrow want probability going to 0
- Sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



Best w?

- Maximum likelihood estimation:

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

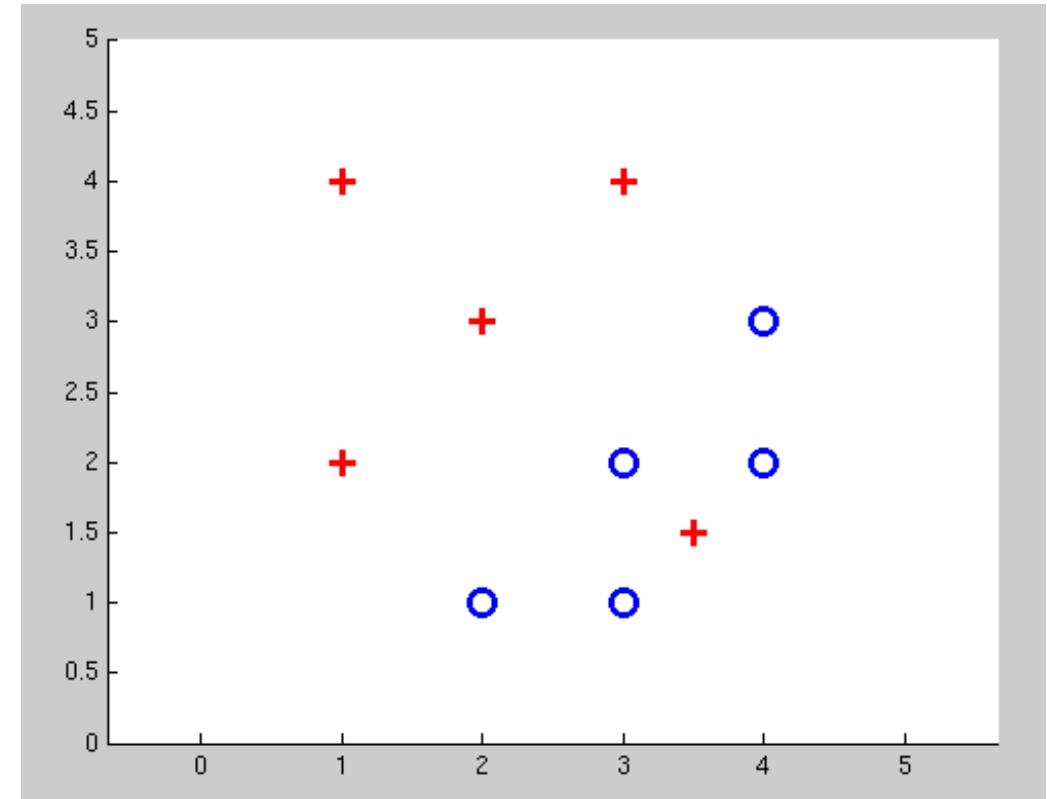
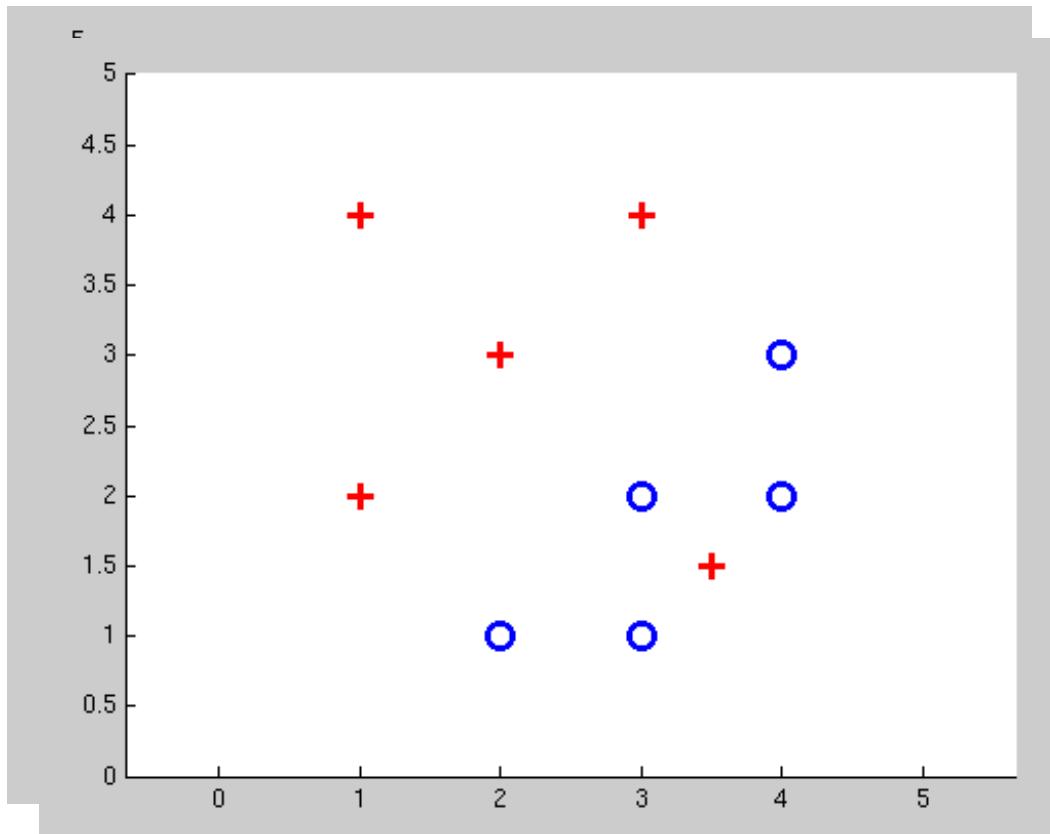
with:

$$P(y^{(i)} = +1 | x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

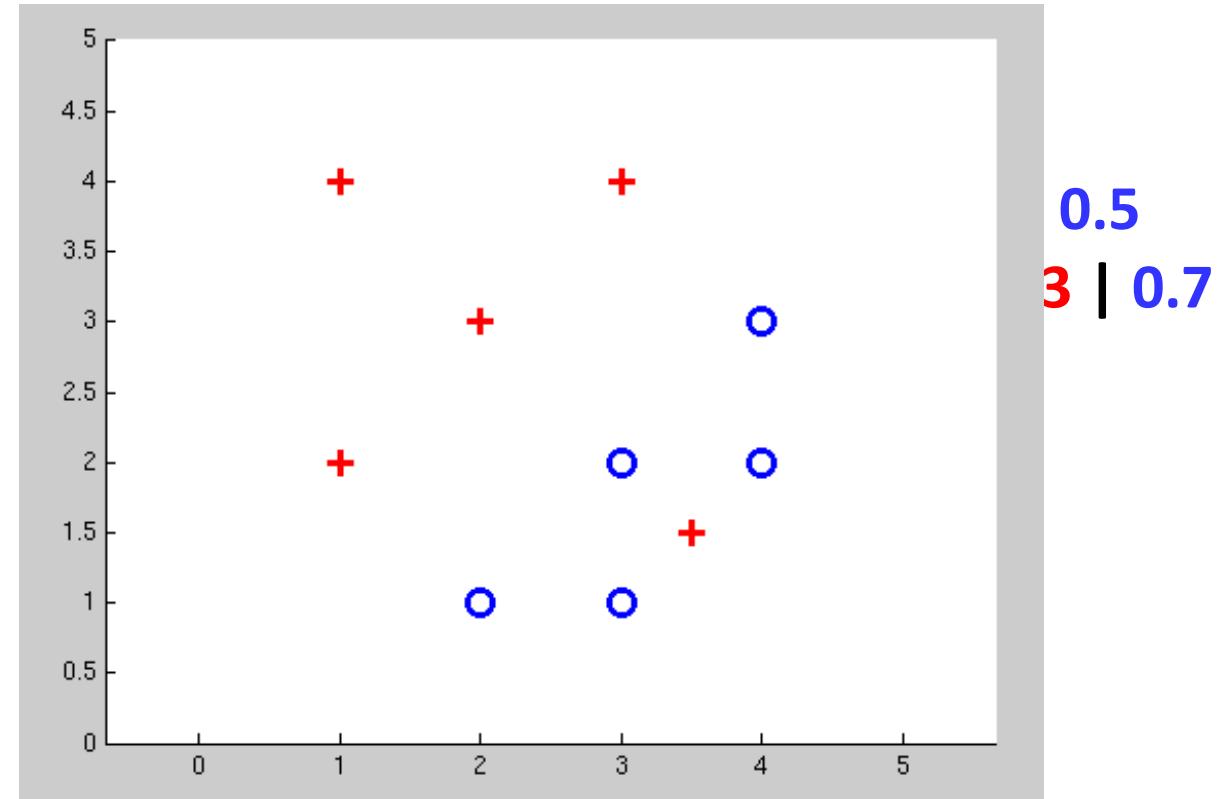
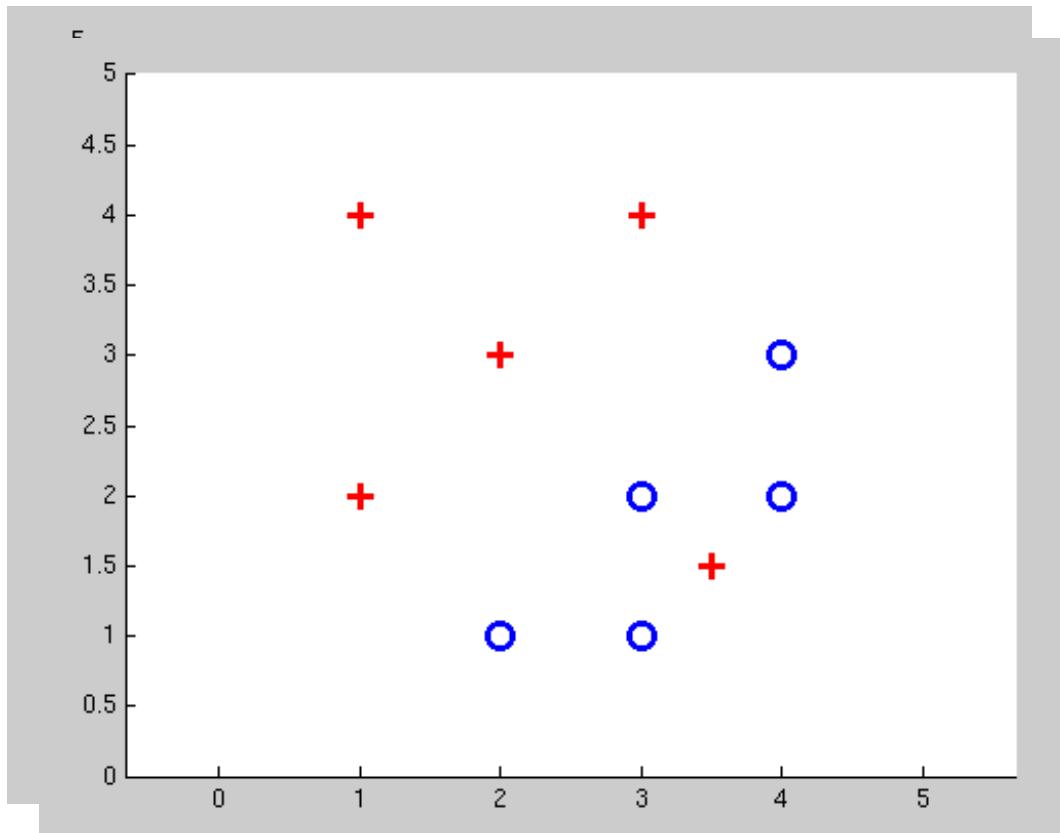
$$P(y^{(i)} = -1 | x^{(i)}; w) = 1 - \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

= Logistic Regression

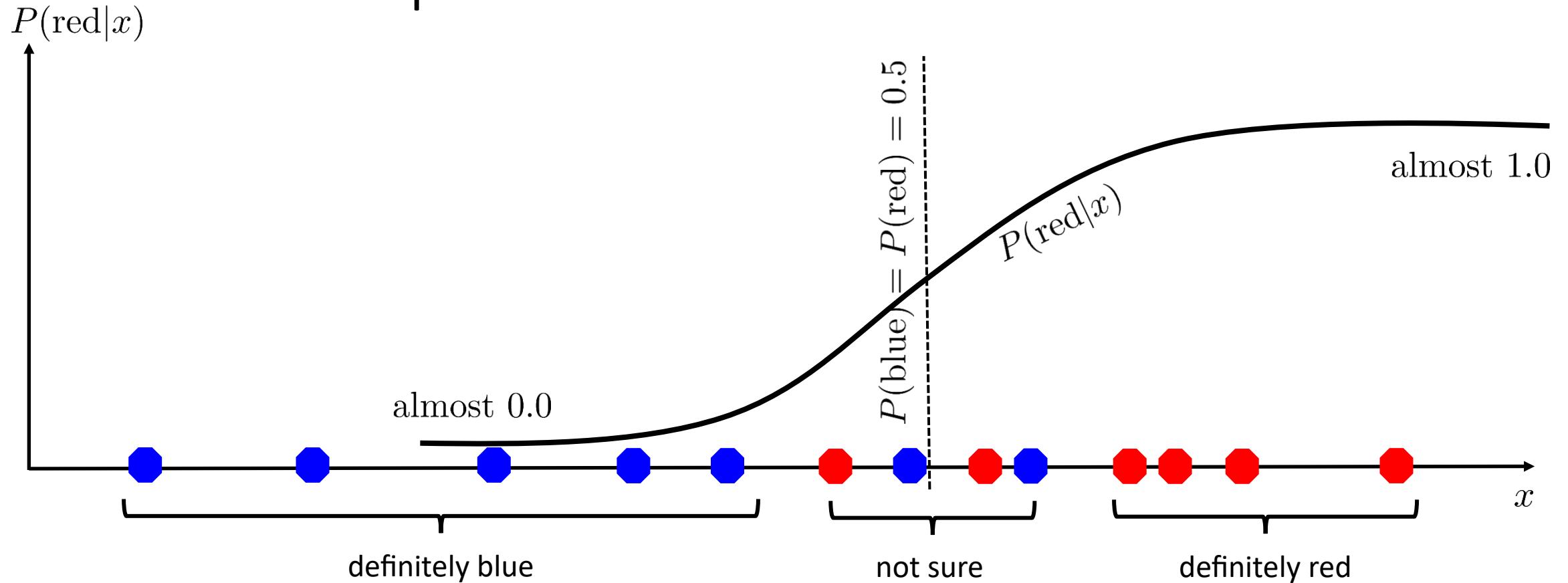
Separable Case: Deterministic Decision – Many Options



Separable Case: Probabilistic Decision – Clear Preference



A 1D Example

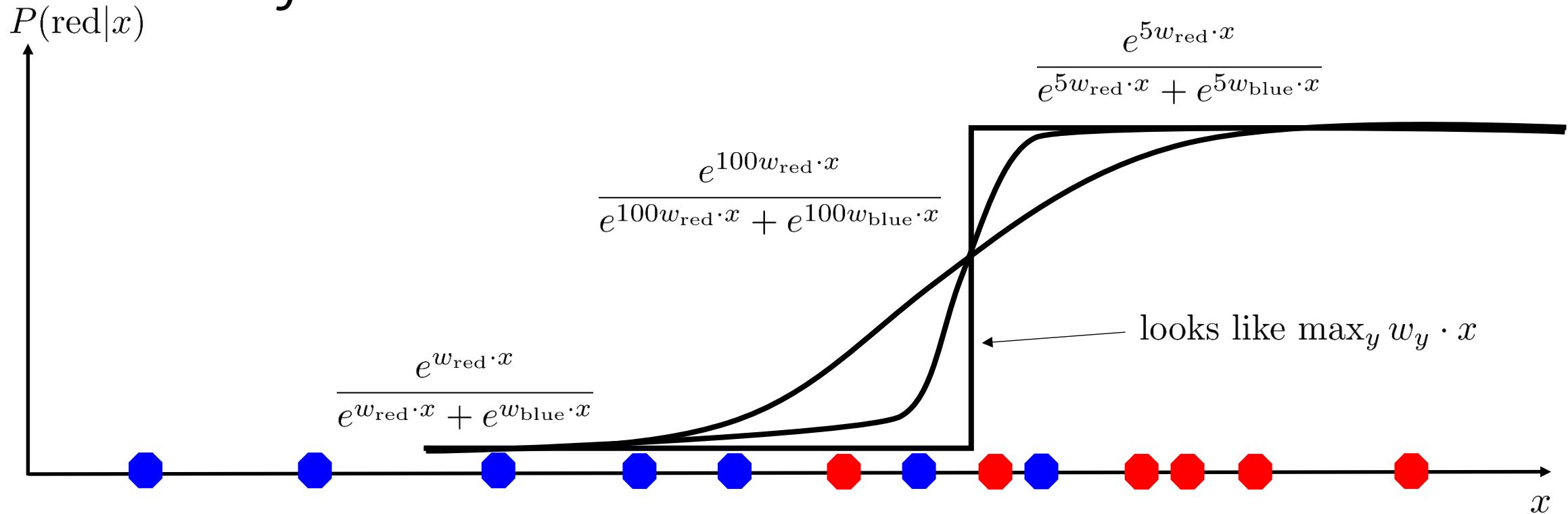


$$P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}}$$

probability increases exponentially as we move away from boundary

normalizer

The Soft Max



$$P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}}$$

Multiclass Logistic Regression

- Recall Perceptron:

- A weight vector for each class:

w_y

- Score (activation) of a class y :

$$w_y \cdot f(x)$$

- Prediction highest score wins

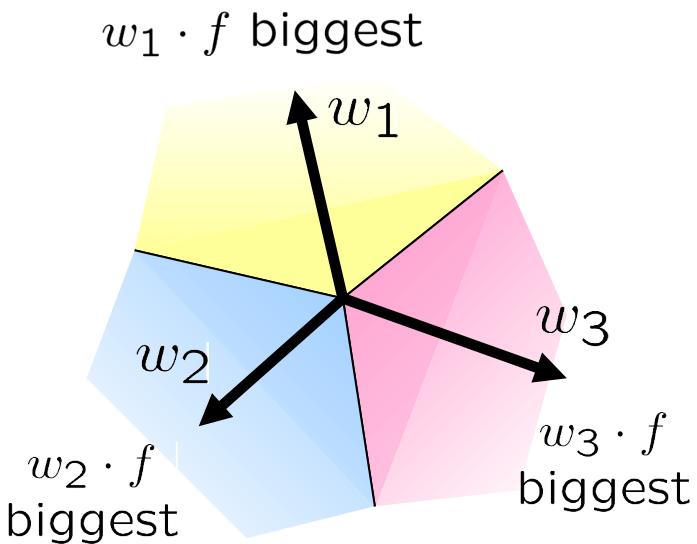
$$y = \arg \max_y w_y \cdot f(x)$$

- How to make the scores into probabilities?

How to make the scores into probabilities:

$$z_1, z_2, z_3 \rightarrow \underbrace{\frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}}_{\text{softmax activations}}$$

original activations



Best w?

- Maximum likelihood estimation:

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

with:

$$P(y^{(i)} | x^{(i)}; w) = \frac{e^{w_y \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$$

= Multi-Class Logistic Regression

- Optimization

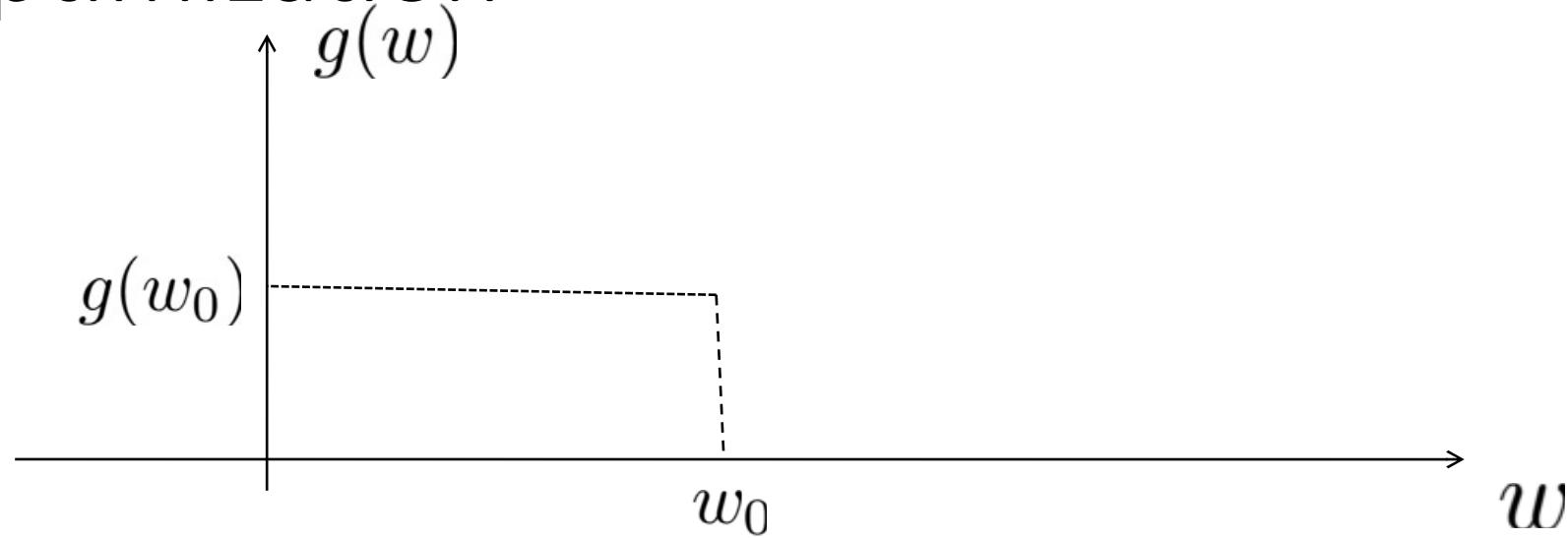
- i.e., how do we solve:

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

Hill Climbing

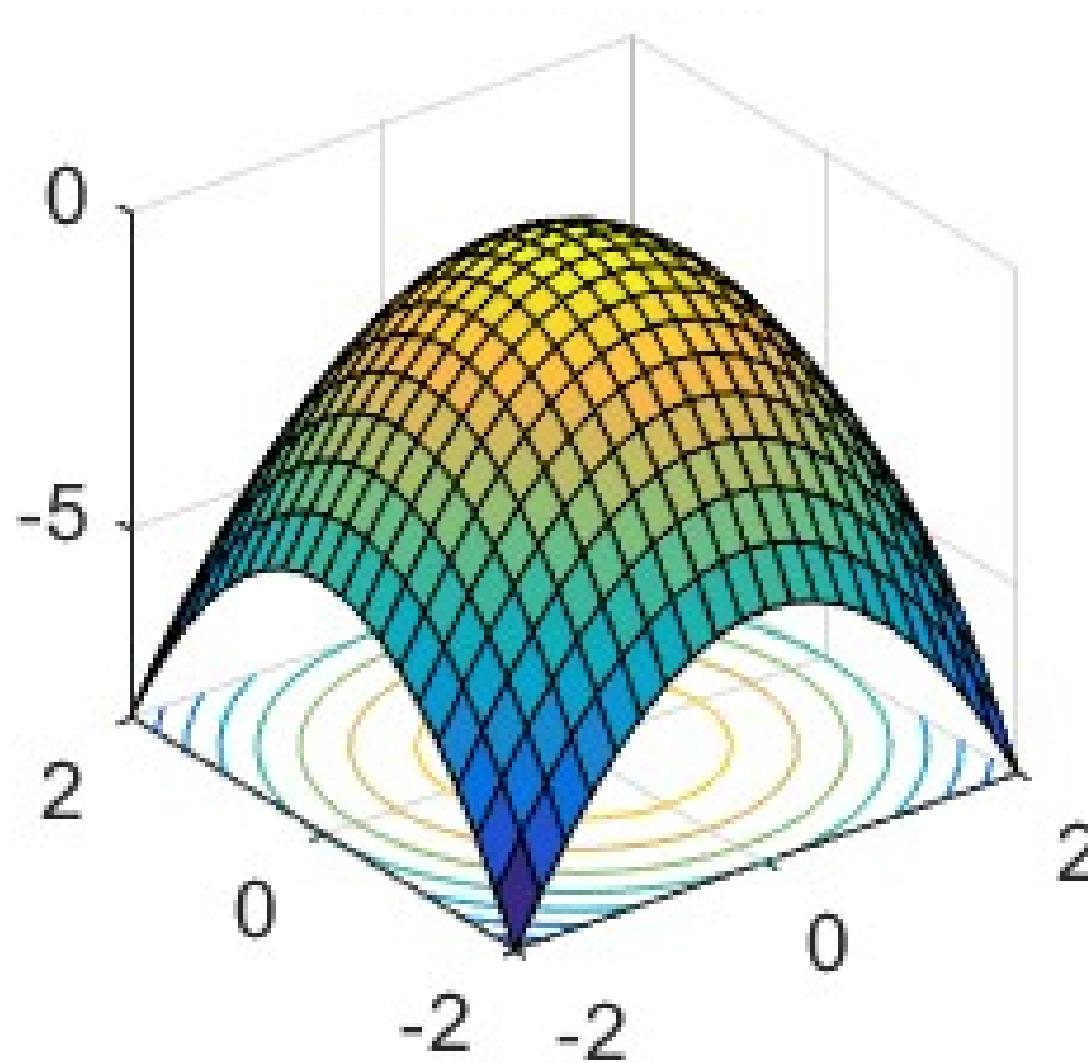
- General idea
 - Start wherever
 - Repeat: move to the best neighboring state
 - If no neighbors better than current, quit
- What's particularly tricky when hill-climbing for multiclass logistic regression?
 - Optimization over a continuous space
 - Infinitely many neighbors!
 - How to do this efficiently?

1-D Optimization



- Could evaluate $g(w_0 + h)$
 - Then step in best direction
- Or, evaluate derivative:
$$\frac{\partial g(w_0)}{\partial w} = \lim_{h \rightarrow 0} \frac{g(w_0 + h) - g(w_0 - h)}{2h}$$
 - Tells which direction to step into

2-D Optimization



Gradient Ascent

- Perform update in uphill direction for each coordinate
- The steeper the slope (i.e. the higher the derivative) the bigger the step for that coordinate
- E.g., consider: $g(w_1, w_2)$

- Updates:

$$w_1 \leftarrow w_1 + \alpha * \frac{\partial g}{\partial w_1}(w_1, w_2)$$

$$w_2 \leftarrow w_2 + \alpha * \frac{\partial g}{\partial w_2}(w_1, w_2)$$

- Updates in vector notation:

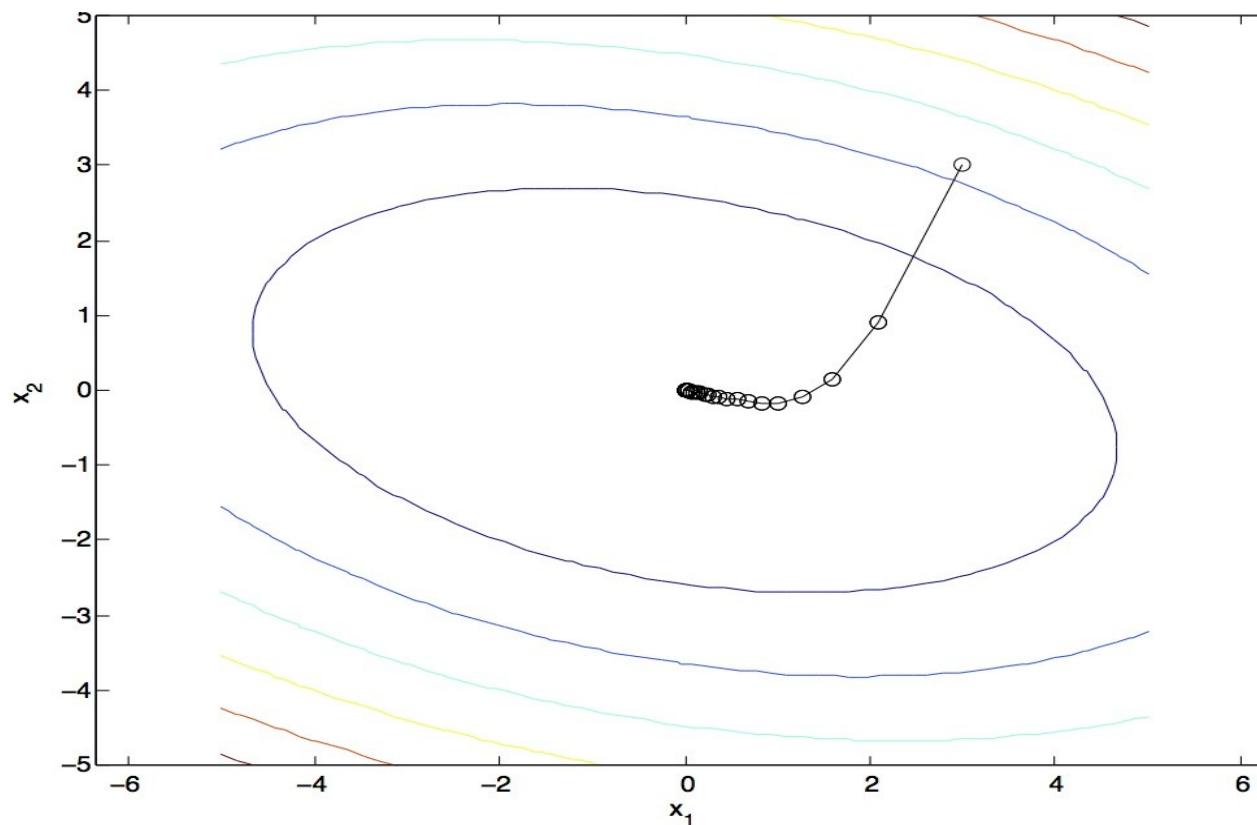
$$w \leftarrow w + \alpha * \nabla_w g(w)$$

with: $\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(w) \\ \frac{\partial g}{\partial w_2}(w) \end{bmatrix}$ = **gradient**

Gradient Ascent

Idea:

- Start somewhere
- Repeat: Take a step in the gradient direction



What is the Steepest Direction?

$$\max_{\Delta: \Delta_1^2 + \Delta_2^2 \leq \varepsilon} g(w + \Delta)$$

- First-Order Taylor Expansion:

$$g(w + \Delta) \approx g(w) + \frac{\partial g}{\partial w_1} \Delta_1 + \frac{\partial g}{\partial w_2} \Delta_2$$

- Steepest Ascent Direction:

$$\max_{\Delta: \Delta_1^2 + \Delta_2^2 \leq \varepsilon} g(w) + \frac{\partial g}{\partial w_1} \Delta_1 + \frac{\partial g}{\partial w_2} \Delta_2$$

- Recall:

$$\max_{\Delta: \|\Delta\| \leq \varepsilon} \Delta^\top a$$

$$\Delta = \varepsilon \frac{a}{\|a\|}$$

- Hence, solution:

$$\Delta = \varepsilon \frac{\nabla g}{\|\nabla g\|}$$

Gradient direction = steepest direction!

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \end{bmatrix}$$

Gradient in n dimensions

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \\ \vdots \\ \frac{\partial g}{\partial w_n} \end{bmatrix}$$

Optimization Procedure: Gradient Ascent

- init w
- for iter = 1, 2, ...

$$w \leftarrow w + \alpha * \nabla g(w)$$

- α : learning rate --- tweaking parameter that needs to be chosen carefully
- How? Try multiple choices
 - Crude rule of thumb: update changes w about 0.1 – 1 %

Batch Gradient Ascent on the Log Likelihood Objective

$$\max_w \text{ll}(w) = \max_w \underbrace{\sum_i \log P(y^{(i)}|x^{(i)}; w)}_{g(w)}$$

- init w
- for iter = 1, 2, ...

$$w \leftarrow w + \alpha * \sum_i \nabla \log P(y^{(i)}|x^{(i)}; w)$$

What will gradient ascent do?

$$w \leftarrow w + \alpha * \sum_i \nabla \log P(y^{(i)} | x^{(i)}; w)$$

$$P(y^{(i)} | x^{(i)}; w) = \frac{e^{w_{y^{(i)}} \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$$

$$\nabla w_{y^{(i)}} f(x^{(i)}) - \nabla \log \sum_y e^{w_y f(x^{(i)})}$$

.....

class weights

$$\frac{1}{\sum_y e^{w_y f(x^{(i)})}} \sum_y \left(e^{w_y f(x^{(i)})} [0^T f(x^{(i)})^T 0^T]^T \right)$$

$$\text{for } y' \text{ weights: } \frac{1}{\sum_y e^{w_y f(x^{(i)})}} e^{w_{y'} f(x^{(i)})} f(x^{(i)})$$

$P(y' | x^{(i)}; w) f(x^{(i)})$ subtracts f from y' weights in proportion to the probability current weights give to y'

Stochastic Gradient Ascent on the Log Likelihood Objective

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

Observation: once gradient on one training example has been computed, might as well incorporate before computing next one

- init w
- for iter = 1, 2, ...
 - pick random j

$$w \leftarrow w + \alpha * \nabla \log P(y^{(j)}|x^{(j)}; w)$$

Mini-Batch Gradient Ascent on the Log Likelihood Objective

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

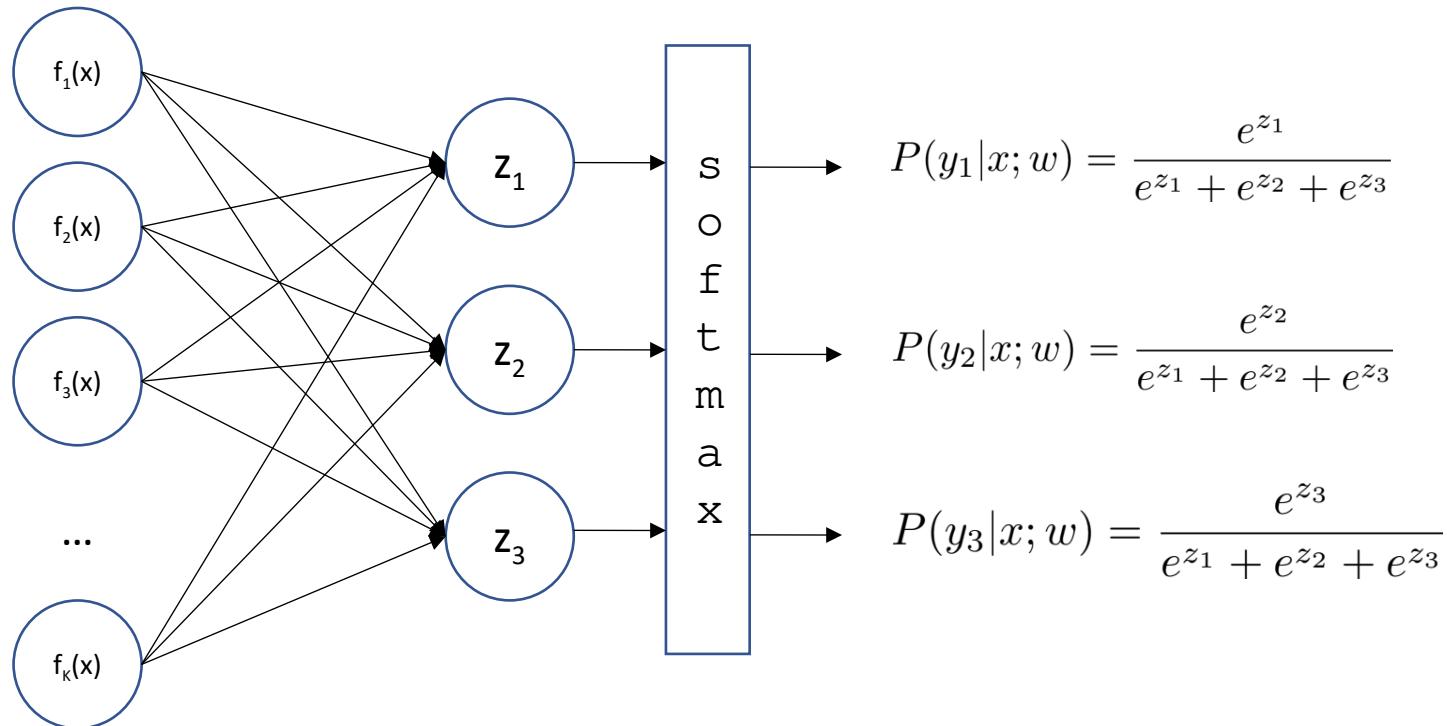
Observation: gradient over small set of training examples (=mini-batch) can be computed in parallel, might as well do that instead of a single one

- init w
- for iter = 1, 2, ...
 - pick random subset of training examples J

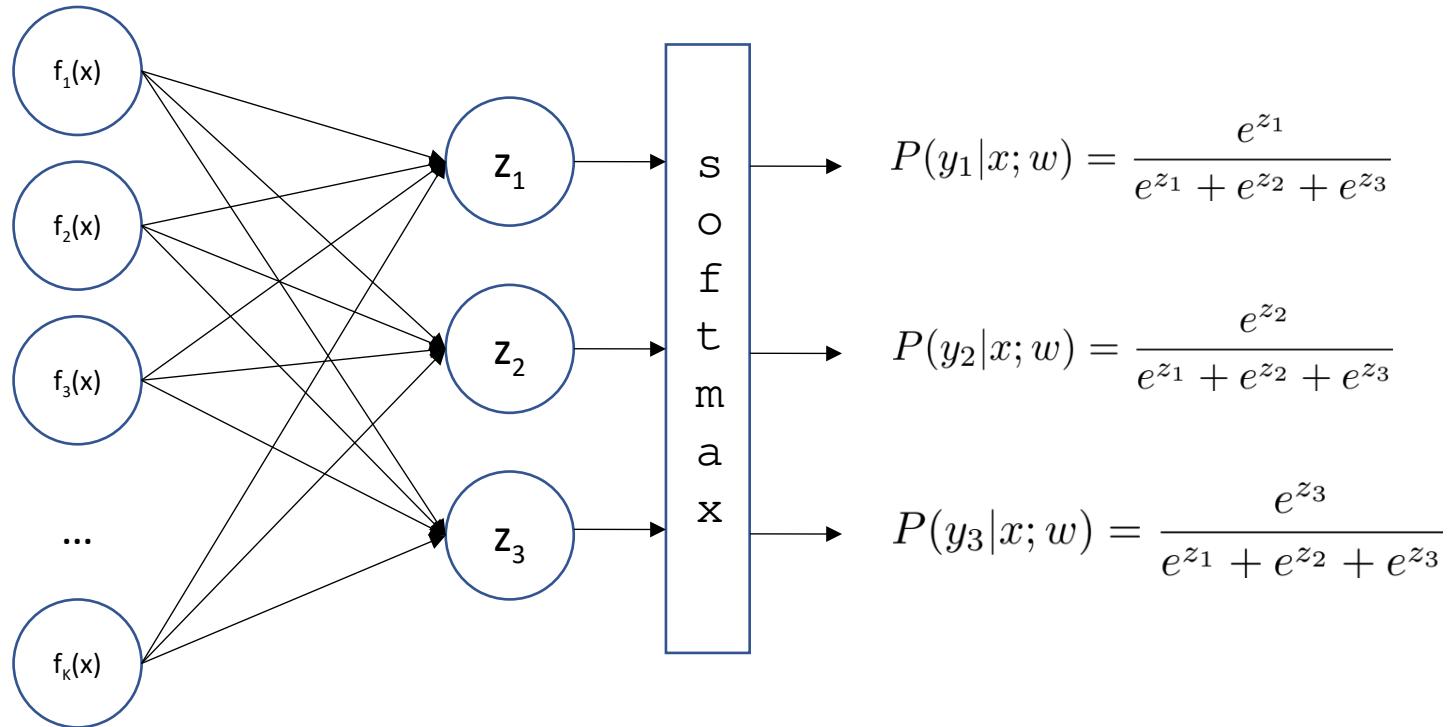
$$w \leftarrow w + \alpha * \sum_{j \in J} \nabla \log P(y^{(j)}|x^{(j)}; w)$$

Multi-class Logistic Regression

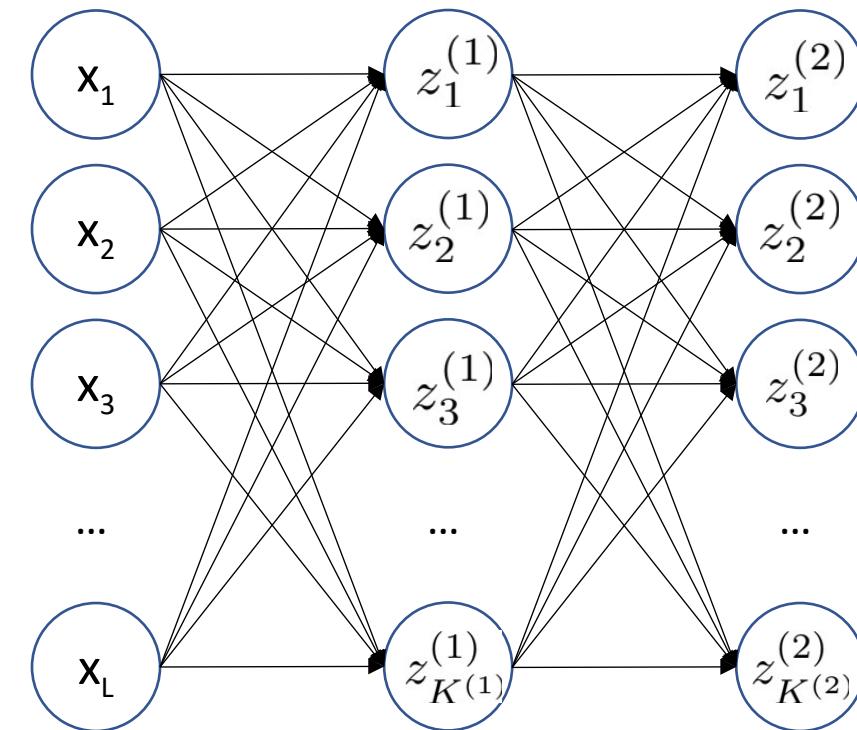
- = special case of neural network



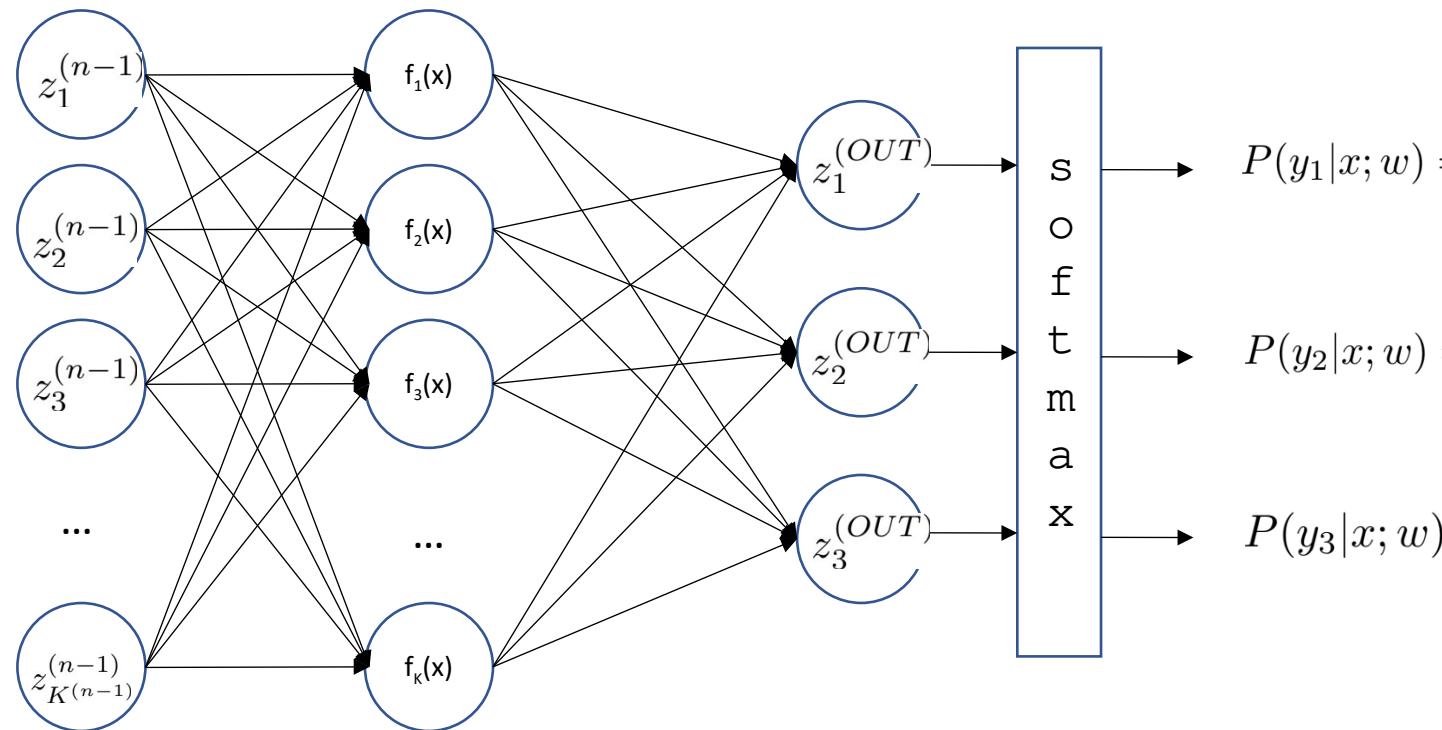
Deep Neural Network = Also learn the features!



Deep Neural Network = Also learn the features!



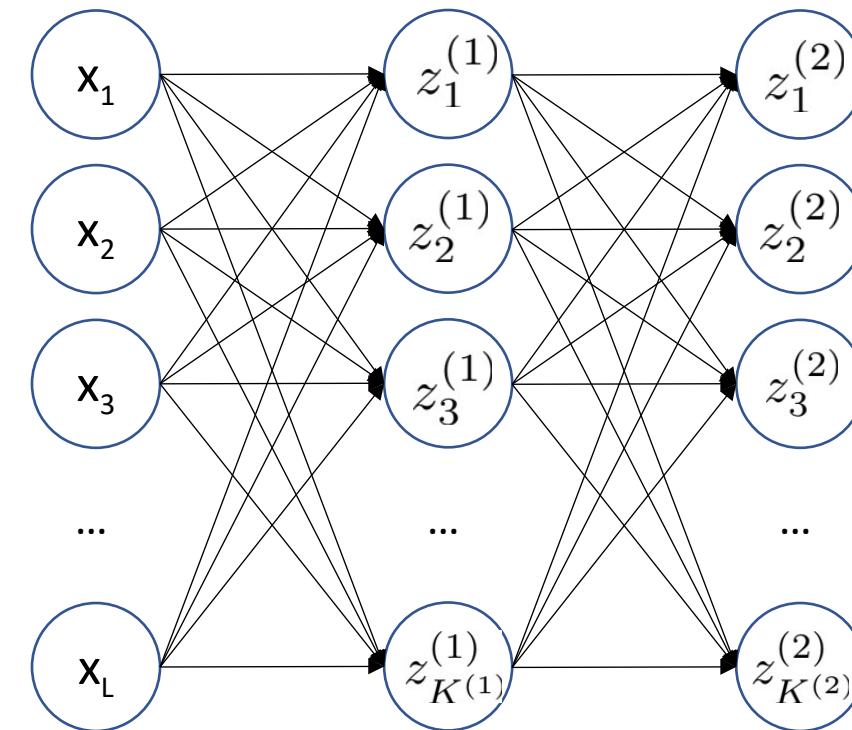
...



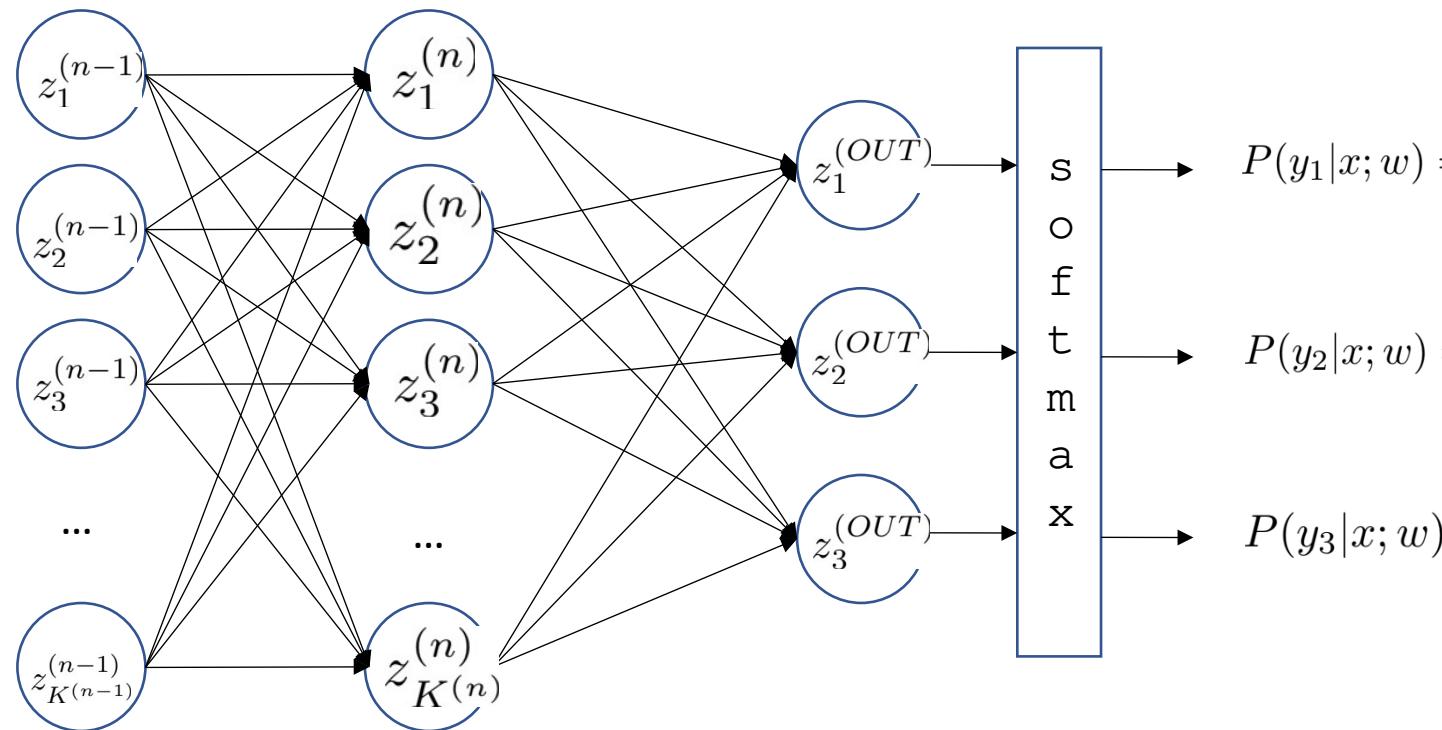
$$z_i^{(k)} = g\left(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)}\right)$$

g = nonlinear activation function

Deep Neural Network = Also learn the features!



...

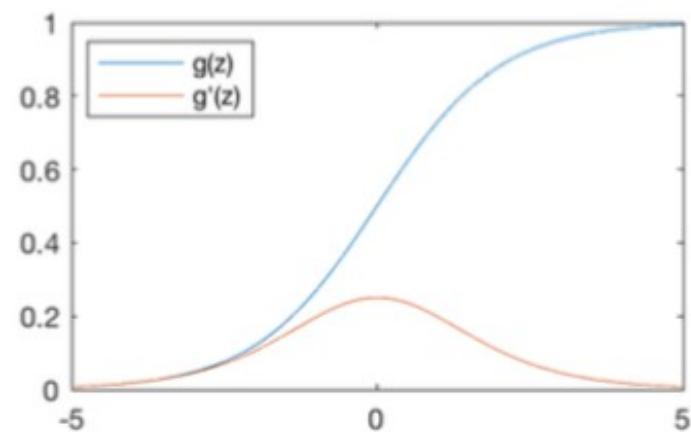


$$z_i^{(k)} = g\left(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)}\right)$$

g = nonlinear activation function

Common Activation Functions

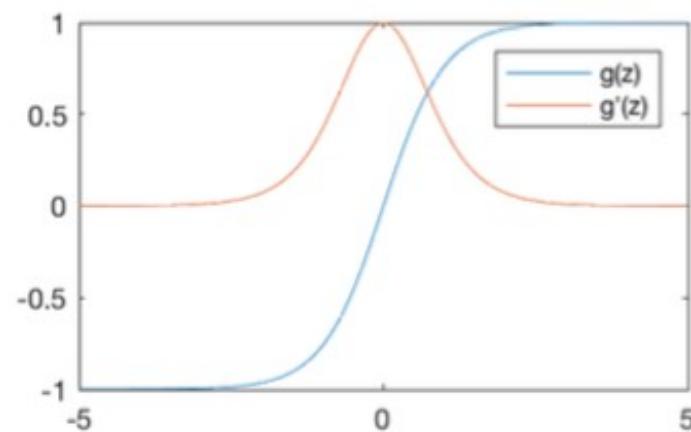
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

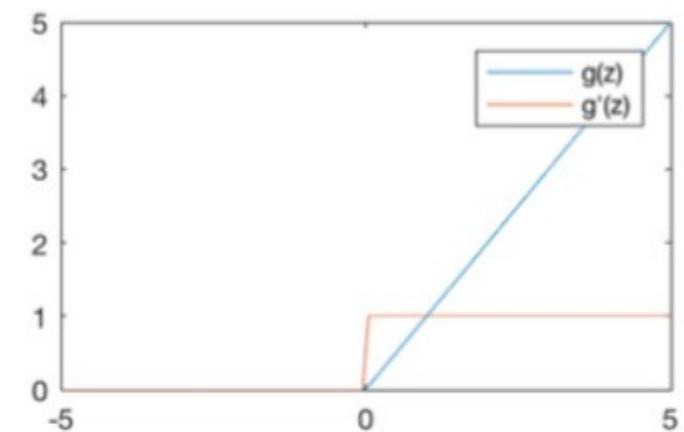
Hyperbolic Tangent



$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Deep Neural Network: Also Learn the Features!

- Training the deep neural network is just like logistic regression:
just w tends to be a much, much larger vector ☺

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

→ just run gradient ascent
+ stop when log likelihood of hold-out data starts to decrease

Neural Networks Properties

- Theorem (Universal Function Approximators). A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.
- Practical considerations
 - Can be seen as learning the features
 - Large number of neurons
 - Danger for overfitting
 - (hence early stopping!)

Universal Function Approximation Theorem*

Hornik theorem 1: Whenever the activation function is *bounded and nonconstant*, then, for any finite measure μ , standard multilayer feedforward networks can approximate any function in $L^p(\mu)$ (the space of all functions on R^k such that $\int_{R^k} |f(x)|^p d\mu(x) < \infty$) arbitrarily well, provided that sufficiently many hidden units are available.

Hornik theorem 2: Whenever the activation function is *continuous, bounded and non-constant*, then, for arbitrary compact subsets $X \subseteq R^k$, standard multilayer feedforward networks can approximate any continuous function on X arbitrarily well with respect to uniform distance, provided that sufficiently many hidden units are available.

- **In words:** Given any continuous function $f(x)$, if a 2-layer neural network has enough hidden units, then there is a choice of weights that allow it to closely approximate $f(x)$.

Cybenko (1989) "Approximations by superpositions of sigmoidal functions"

Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks"

Leshno and Schocken (1991) "Multilayer Feedforward Networks with Non-Polynomial Activation Functions Can Approximate Any Function"

Universal Function Approximation Theorem*

Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control,
Signals, and Systems
© 1989 Springer-Verlag New York Inc.

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

1. Introduction

A number of diverse application areas are concerned with the representation of general functions of an n -dimensional real variable, $x \in \mathbb{R}^n$, by finite linear combinations of the form

$$\sum_{j=1}^N \alpha_j \sigma(y_j^T x + \theta_j), \quad (1)$$

where $y_j \in \mathbb{R}^n$ and $\alpha_j, \theta \in \mathbb{R}$ are fixed. (y^T is the transpose of y so that $y^T x$ is the inner product of y and x). Here the univariate function σ depends heavily on the context of the application. Our major concern is with so-called sigmoidal σ 's:

$$\sigma(t) \rightarrow \begin{cases} 1 & \text{as } t \rightarrow +\infty, \\ 0 & \text{as } t \rightarrow -\infty. \end{cases}$$

Such functions arise naturally in neural network theory as the activation function of a neural node (or *unit* as is becoming the preferred term) [L1], [RHM]. The main result of this paper is a demonstration of the fact that sums of the form (1) are dense in the space of continuous functions on the unit cube if σ is any continuous sigmoidal

* Date received: October 21, 1988. Date revised: February 17, 1989. This research was supported in part by NSF Grant DCR-8619103, ONR Contract N000-86-G-0202 and DOE Grant DE-FG02-85ER25001.

† Center for Supercomputing Research and Development and Department of Electrical and Computer Engineering, University of Illinois, Urbana, Illinois 61801, U.S.A.

Neural Networks, Vol. 4, pp. 251–257, 1991
Printed in the USA. All rights reserved.
(893-6680/91) \$3.00 + .00
Copyright © 1991 Pergamon Press plc

ORIGINAL CONTRIBUTION

Approximation Capabilities of Multilayer Feedforward Networks

KURT HORNIK

Technische Universität Wien, Vienna, Austria

(Received 30 January 1990; revised and accepted 25 October 1990)

Abstract—We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^p(\mu)$ performance criteria, for arbitrary finite input environment measures μ , provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.

Keywords—Multilayer feedforward networks, Activation function, Universal approximation capabilities, Input environment measure, $L^p(\mu)$ approximation, Uniform approximation, Sobolev spaces, Smooth approximation.

1. INTRODUCTION

The approximation capabilities of neural network architectures have recently been investigated by many authors, including Carroll and Dickinson (1989), Cybenko (1989), Funahashi (1989), Gallant and White (1988), Hecht-Nielsen (1989), Hornik, Stinchcombe, and White (1989, 1990), Itoh and Miyake (1988), Lapedes and Farber (1988), Stinchcombe and White (1989, 1990). (This list is by no means complete.)

If we think of the network architecture as a rule for computing values at l output units given values at k input units, hence implementing a class of mappings from \mathbb{R}^k to \mathbb{R}^l , we can ask how well arbitrary mappings from \mathbb{R}^k to \mathbb{R}^l can be approximated by the network, in particular, if as many hidden units as required for internal representation and computation may be employed.

How to measure the accuracy of approximation depends on how we measure closeness between functions, which in turn varies significantly with the specific problem to be dealt with. In many applications, it is necessary to have the network perform simultaneously well on all input samples taken from some compact input set X in \mathbb{R}^k . In this case, closeness is

measured by the uniform distance between functions on X , that is,

$$\rho_{\infty, X}(f, g) = \sup_{x \in X} |f(x) - g(x)|.$$

In other applications, we think of the inputs as random variables and are interested in the *average performance* where the average is taken with respect to the input environment measure μ , where $\mu(\mathbb{R}^k) < \infty$. In this case, closeness is measured by the $L^p(\mu)$ distances

$$\rho_{p, X}(f, g) = \left[\int_{\mathbb{R}^k} |f(x) - g(x)|^p d\mu(x) \right]^{1/p}.$$

$1 \leq p < \infty$, the most popular choice being $p = 2$, corresponding to mean square error.

Of course, there are many more ways of measuring closeness of functions. In particular, in many applications, it is also necessary that the *derivatives* of the approximating function implemented by the network closely resemble those of the function to be approximated, up to some order. This issue was first taken up in Hornik et al. (1990), who discuss the sources of need of smooth functional approximation in more detail. Typical examples arise in robotics (learning of smooth movements) and signal processing (analysis of chaotic time series); for a recent application to problems of nonparametric inference in statistics and econometrics, see Gallant and White (1989).

All papers establishing certain approximation ca-

Requests for reprints should be sent to Kurt Hornik, Institut für Statistik und Wahrscheinlichkeitstheorie, Technische Universität Wien, Wiedner Hauptstraße 8-10/107, A-1040 Wien, Austria.

MULTILAYER FEEDFORWARD NETWORKS WITH NON-POLYNOMIAL ACTIVATION FUNCTIONS CAN APPROXIMATE ANY FUNCTION

by

Moshe Leshno
Faculty of Management
Tel Aviv University
Tel Aviv, Israel 69978

and

Shimon Schocken
Leonard N. Stern School of Business
New York University
New York, NY 10003

September 1991

Center for Research on Information Systems
Information Systems Department
Leonard N. Stern School of Business
New York University

Working Paper Series

STERN IS-91-26

Appeared previously as *Working Paper No. 21/91* at The Israel Institute Of Business Research

Cybenko (1989) "Approximations by superpositions of sigmoidal functions"

Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks"

Leshno and Schocken (1991) "Multilayer Feedforward Networks with Non-Polynomial Activation Functions Can Approximate Any Function"

How about computing all the derivatives?

- Derivatives tables:

$$\frac{d}{dx}(a) = 0$$

$$\frac{d}{dx}(x) = 1$$

$$\frac{d}{dx}(au) = a \frac{du}{dx}$$

$$\frac{d}{dx}(u+v-w) = \frac{du}{dx} + \frac{dv}{dx} - \frac{dw}{dx}$$

$$\frac{d}{dx}(uv) = u \frac{dv}{dx} + v \frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{u}{v}\right) = \frac{1}{v} \frac{du}{dx} - \frac{u}{v^2} \frac{dv}{dx}$$

$$\frac{d}{dx}(u^n) = nu^{n-1} \frac{du}{dx}$$

$$\frac{d}{dx}(\sqrt{u}) = \frac{1}{2\sqrt{u}} \frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{1}{u}\right) = -\frac{1}{u^2} \frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{1}{u^n}\right) = -\frac{n}{u^{n+1}} \frac{du}{dx}$$

$$\frac{d}{dx}[f(u)] = \frac{d}{du}[f(u)] \frac{du}{dx}$$

$$\frac{d}{dx}[\ln u] = \frac{d}{dx}[\log_e u] = \frac{1}{u} \frac{du}{dx}$$

$$\frac{d}{dx}[\log_a u] = \log_a e \frac{1}{u} \frac{du}{dx}$$

$$\frac{d}{dx}e^u = e^u \frac{du}{dx}$$

$$\frac{d}{dx}a^u = a^u \ln a \frac{du}{dx}$$

$$\frac{d}{dx}(u^v) = vu^{v-1} \frac{du}{dx} + \ln u \ u^v \frac{dv}{dx}$$

$$\frac{d}{dx}\sin u = \cos u \frac{du}{dx}$$

$$\frac{d}{dx}\cos u = -\sin u \frac{du}{dx}$$

$$\frac{d}{dx}\tan u = \sec^2 u \frac{du}{dx}$$

$$\frac{d}{dx}\cot u = -\csc^2 u \frac{du}{dx}$$

$$\frac{d}{dx}\sec u = \sec u \tan u \frac{du}{dx}$$

$$\frac{d}{dx}\csc u = -\csc u \cot u \frac{du}{dx}$$

How about computing all the derivatives?

- But neural net f is never one of those?
 - No problem: CHAIN RULE:

If
$$f(x) = g(h(x))$$

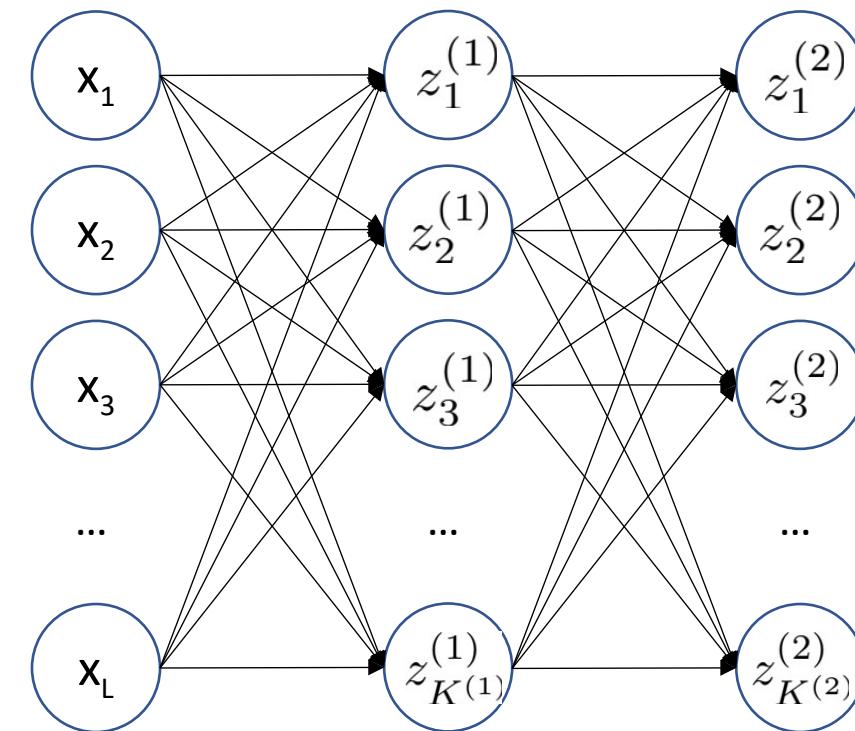
Then
$$f'(x) = g'(h(x))h'(x)$$

→ Derivatives can be computed by following well-defined procedures

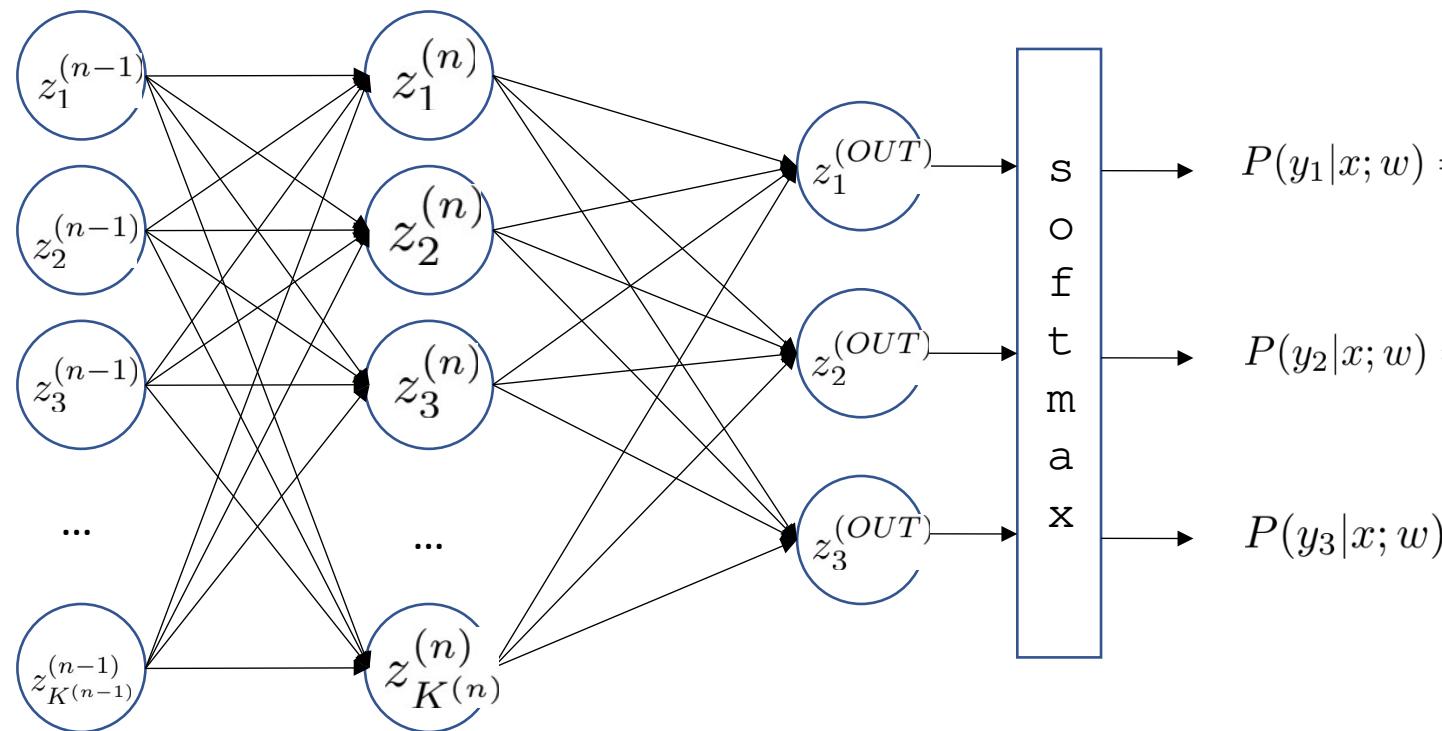
Automatic Differentiation

- Automatic differentiation software
 - e.g. PyTorch, TensorFlow, JAX
 - Only need to program the function $g(x,y,w)$
 - Can automatically compute all derivatives w.r.t. all entries in w
 - This is typically done by caching info during forward computation pass of f , and then doing a backward pass = “backpropagation”
 - Autodiff / Backpropagation can often be done at computational cost comparable to the forward pass

Training a Network (setting weights)



...



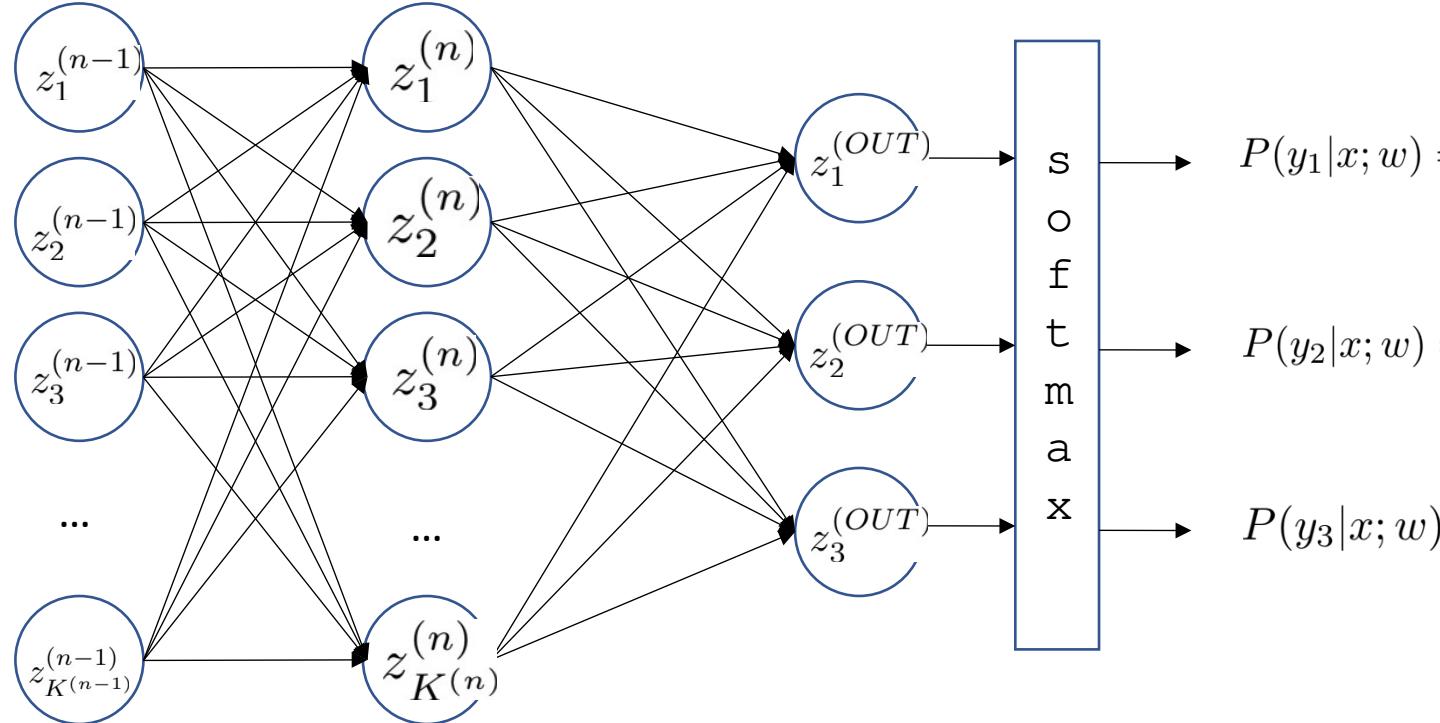
$$z_i^{(k)} = g\left(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)}\right)$$

g = nonlinear activation function

Training a Network

Key words:

- Forward
- Backwards
- Gradient
- Backprop

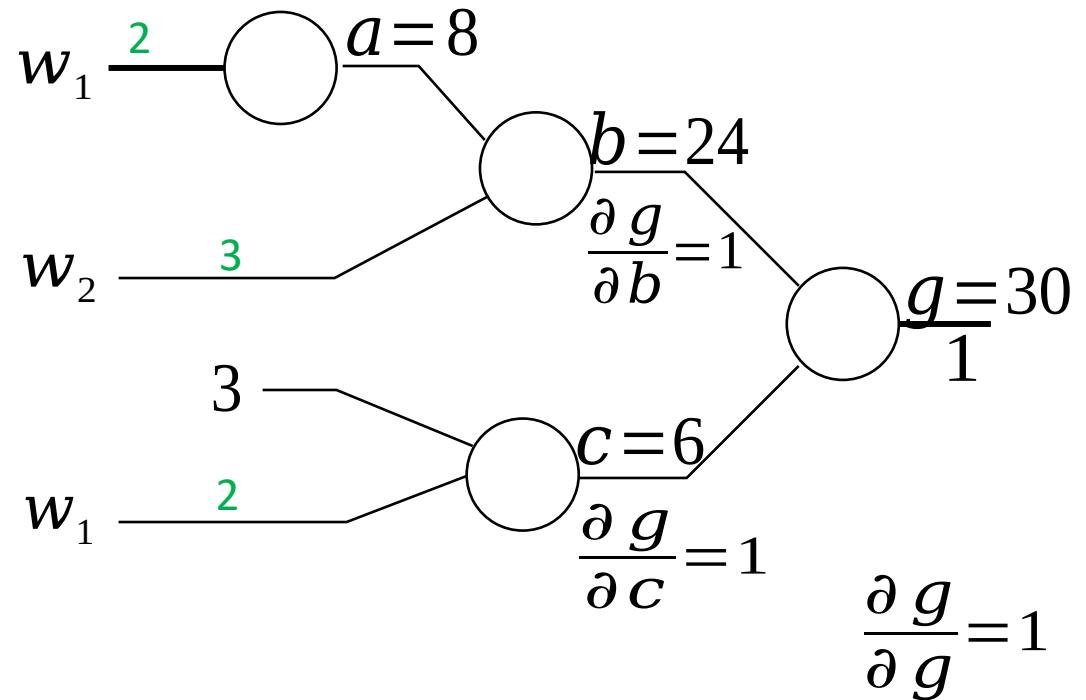


g = nonlinear activation function

Back Propagation:

- Suppose we have a and want the gradient at a
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial a}$.

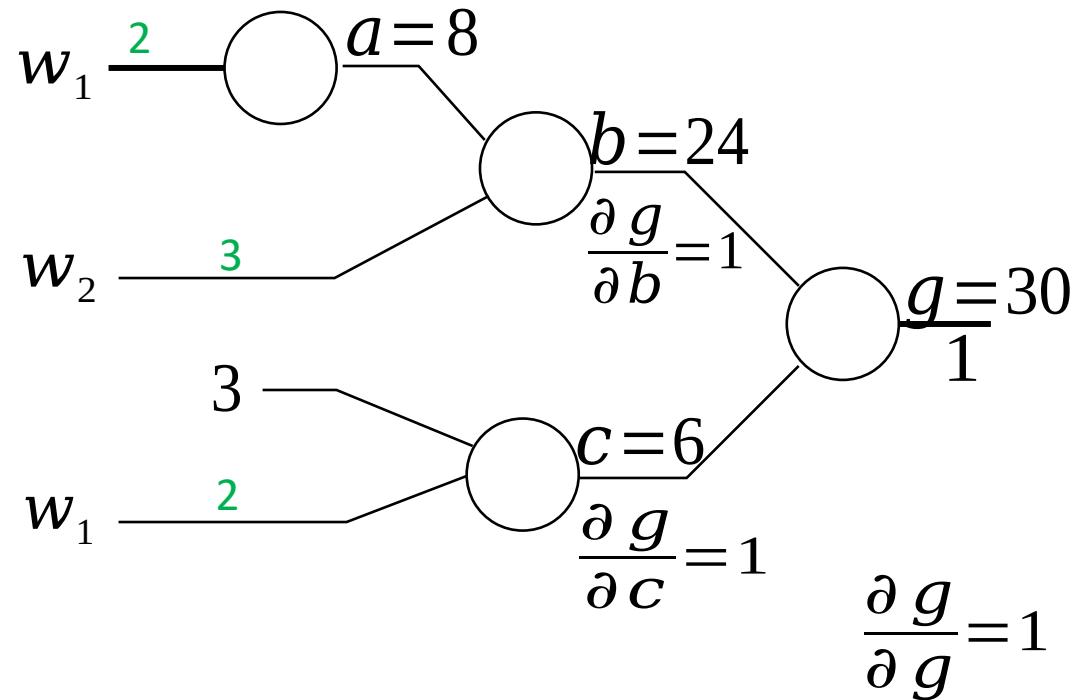
• ,



Back Propagation:

- Suppose we have a and want the gradient at a
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial a}$.

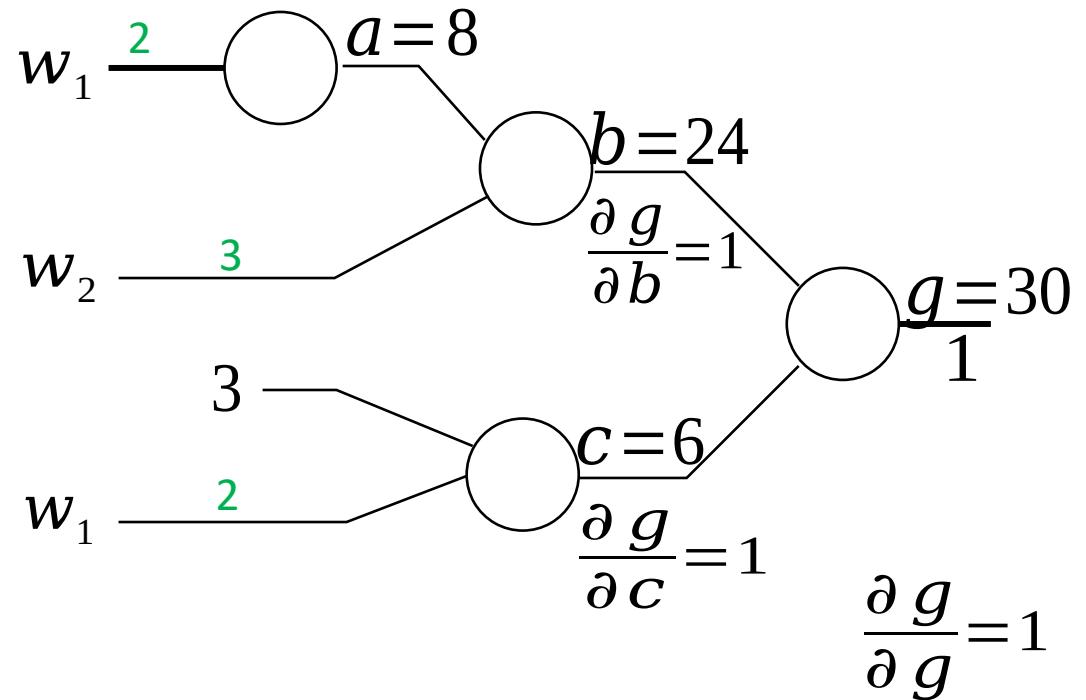
• ,



Back Propagation:

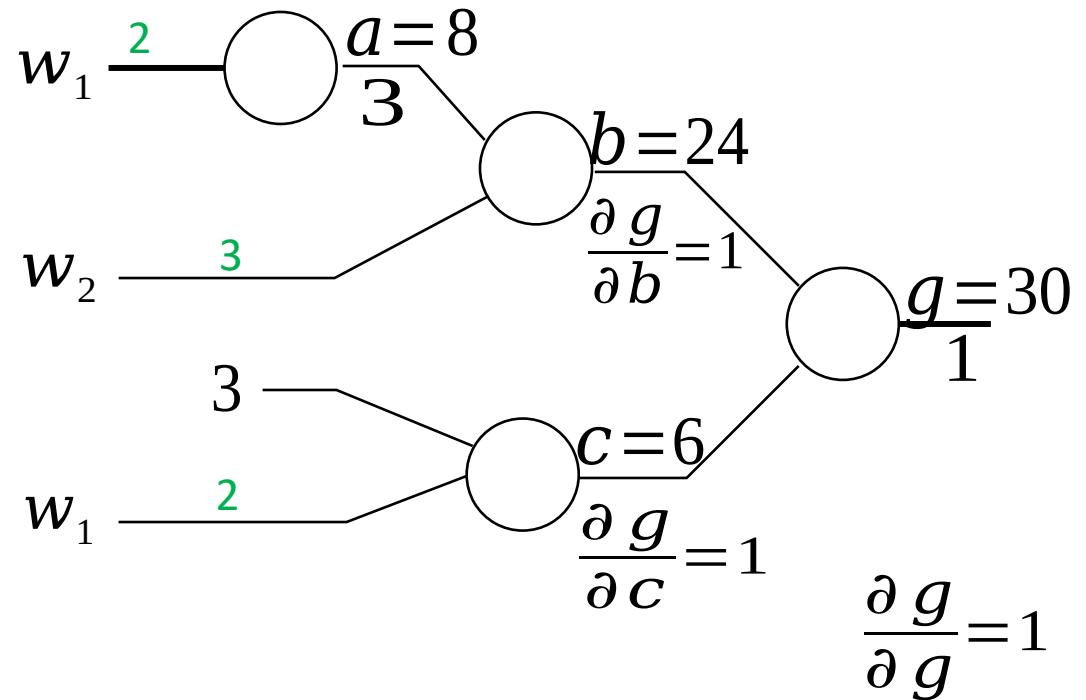
- Suppose we have a and want the gradient at a
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial a}$.

• ,



Back Propagation:

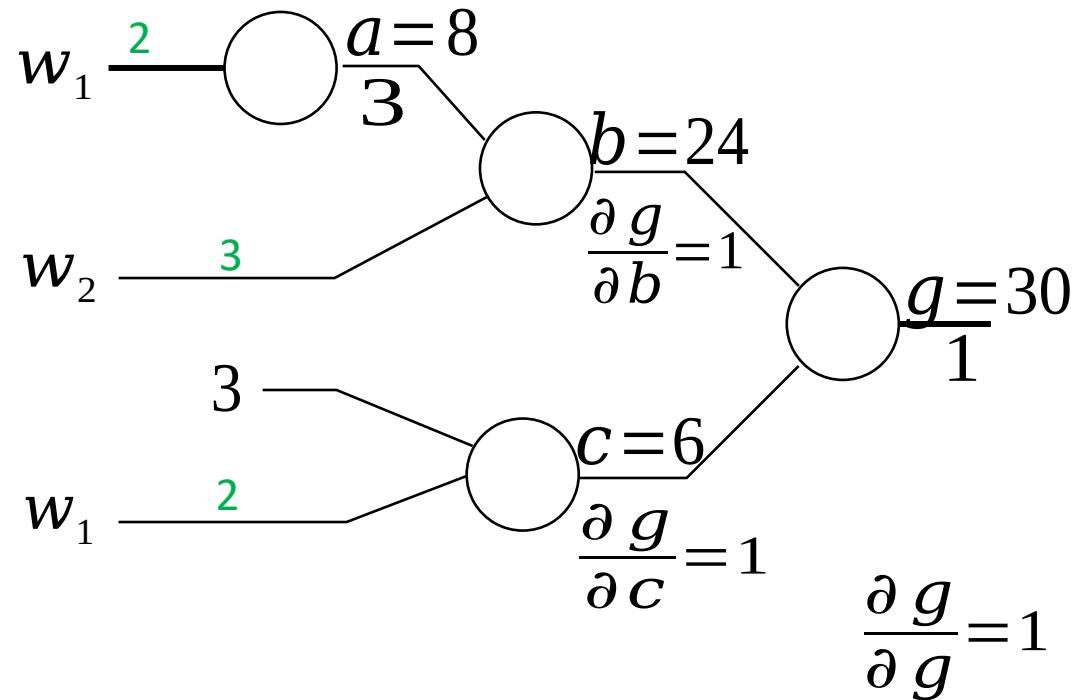
- Suppose we have a and want the gradient at a
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial a}$.
 - ,



Back Propagation:

- Suppose we have a and want the gradient at a
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial a}$.

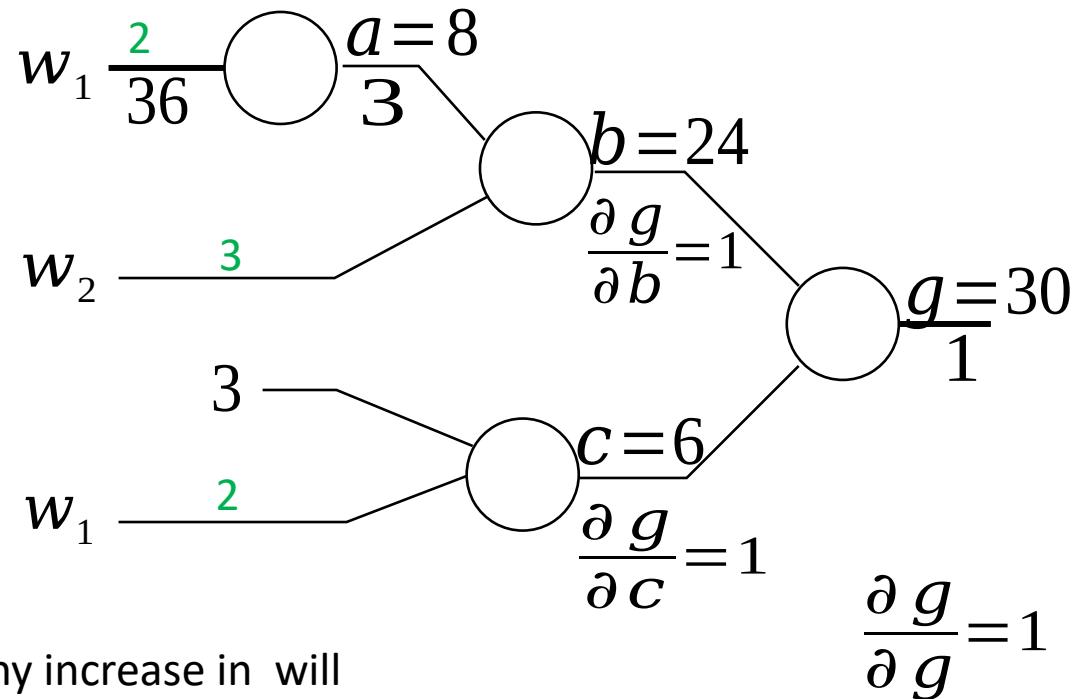
• ,



Back Propagation:

- Suppose we have g and want the gradient at a
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial a}$.

• ,

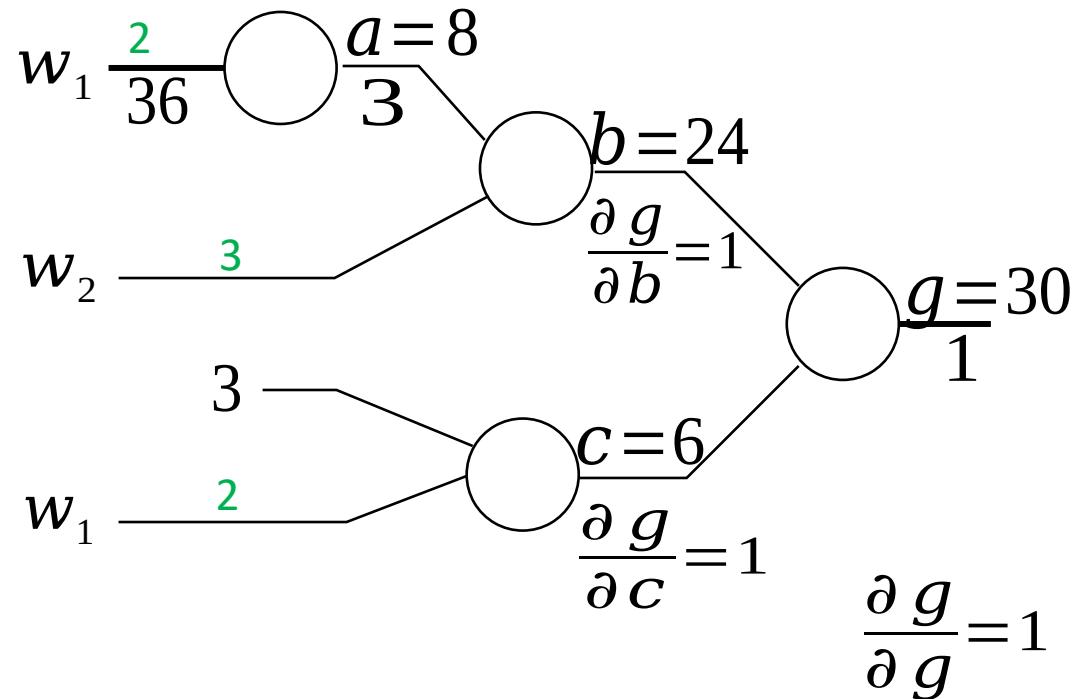


Interpretation: A tiny increase in w_1 will result in an approximately $3 \times 2 = 6$ increase in g due to this cube function.

Back Propagation:

- Suppose we have g and want the gradient at θ
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial \theta}$.
 - ,

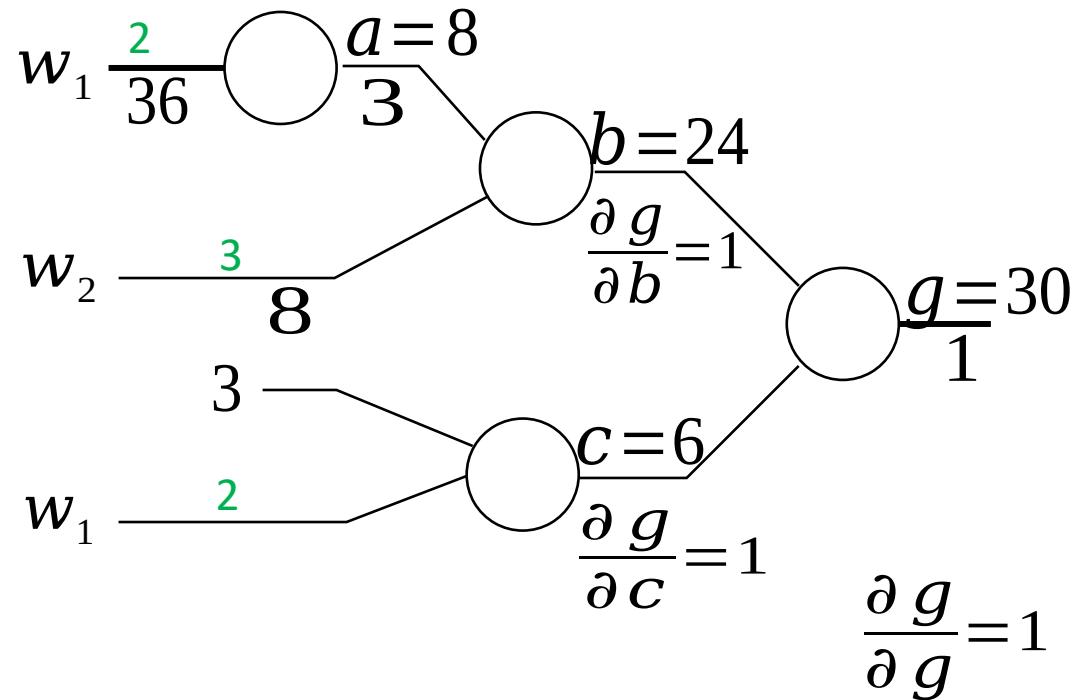
Hint: may be useful.



Back Propagation:

- Suppose we have g and want the gradient at x
- Think of the function as a composition of many functions.
 - Can use derivative chain rule to compute $\frac{\partial g}{\partial x}$.

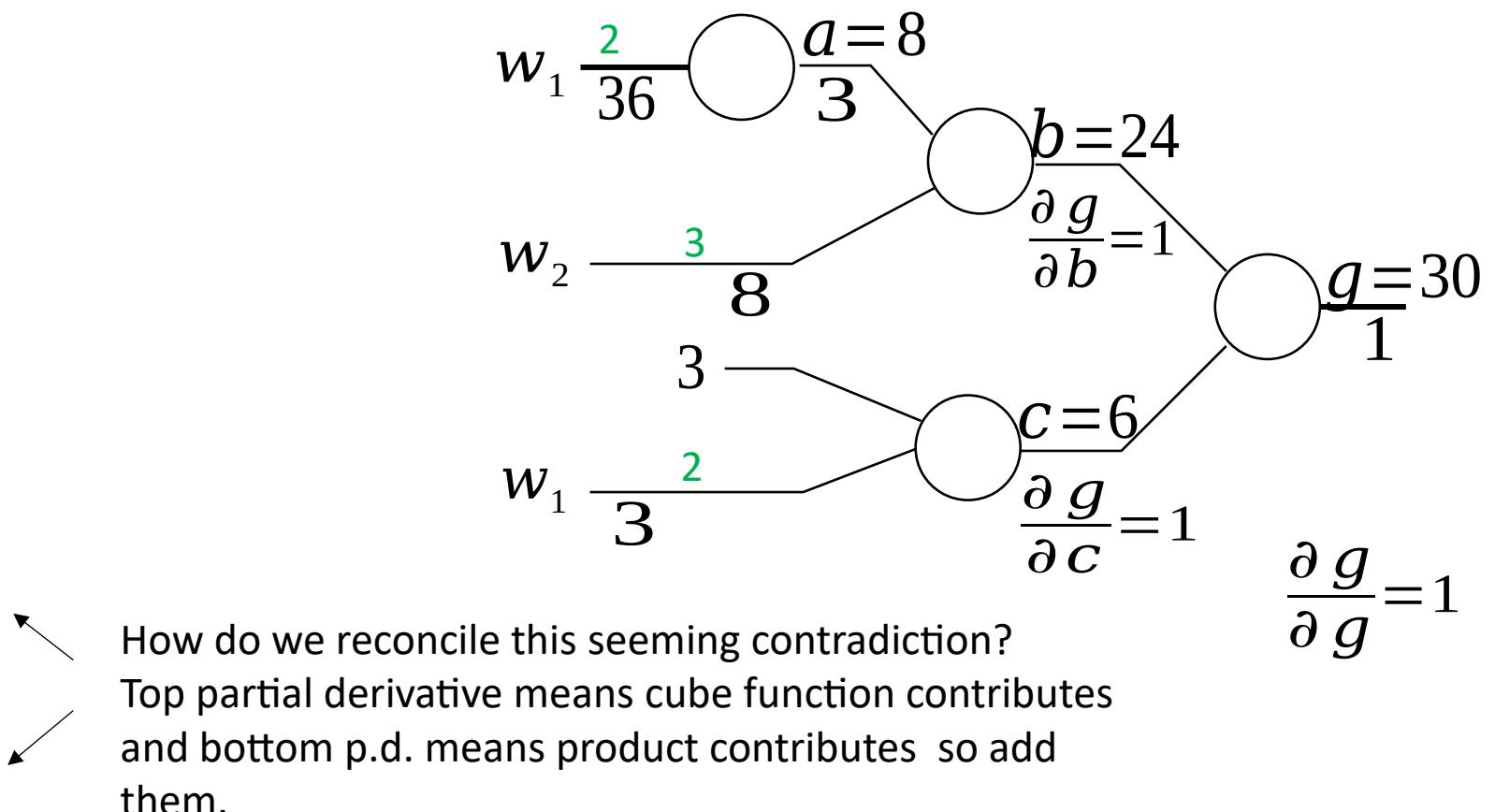
• ,



Back Propagation:

- Suppose we have g and want the gradient at a
- Think of the function as a composition of many functions, use chain rule.

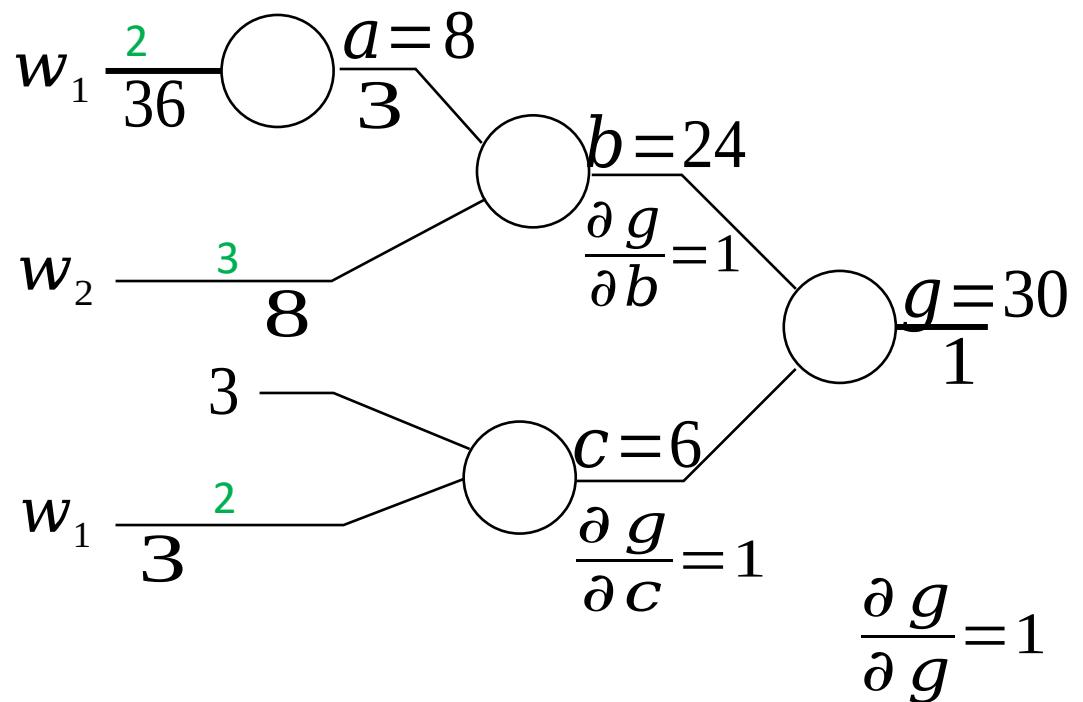
• ,



Back Propagation:

- Suppose we have g and want the gradient at w_1
- Think of the function as a composition of many functions, use chain rule.

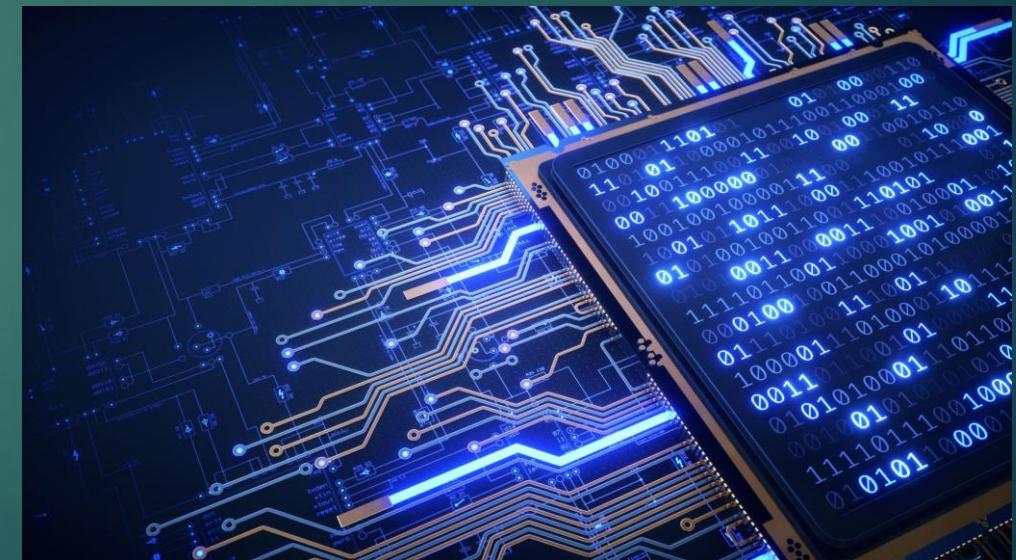
• ,



$$\nabla g = \left[\frac{\partial g}{\partial w_1}, \frac{\partial g}{\partial w_2} \right] = [39, 8]$$

Brain Computer Interaction

Introduction to BCI



What's Today

- ▶ The Course Content
- ▶ What is Brain Computer Interaction
- ▶ History of BCI

Imagine a machine that records
feelings, emotions, even your
hopes and dreams.

And imagine that it can transfer
these experiences from
one mind to another...

B R A I N S T O R M¹⁵

METRO-GOLDWYN-MAYER presents

A J F PRODUCTION

A DOUGLAS TRUMBULL FILM "BRAINSTORM"
CHRISTOPHER WALKEN · NATALIE WOOD
LOUISE FLETCHER · CLIFF ROBERTSON

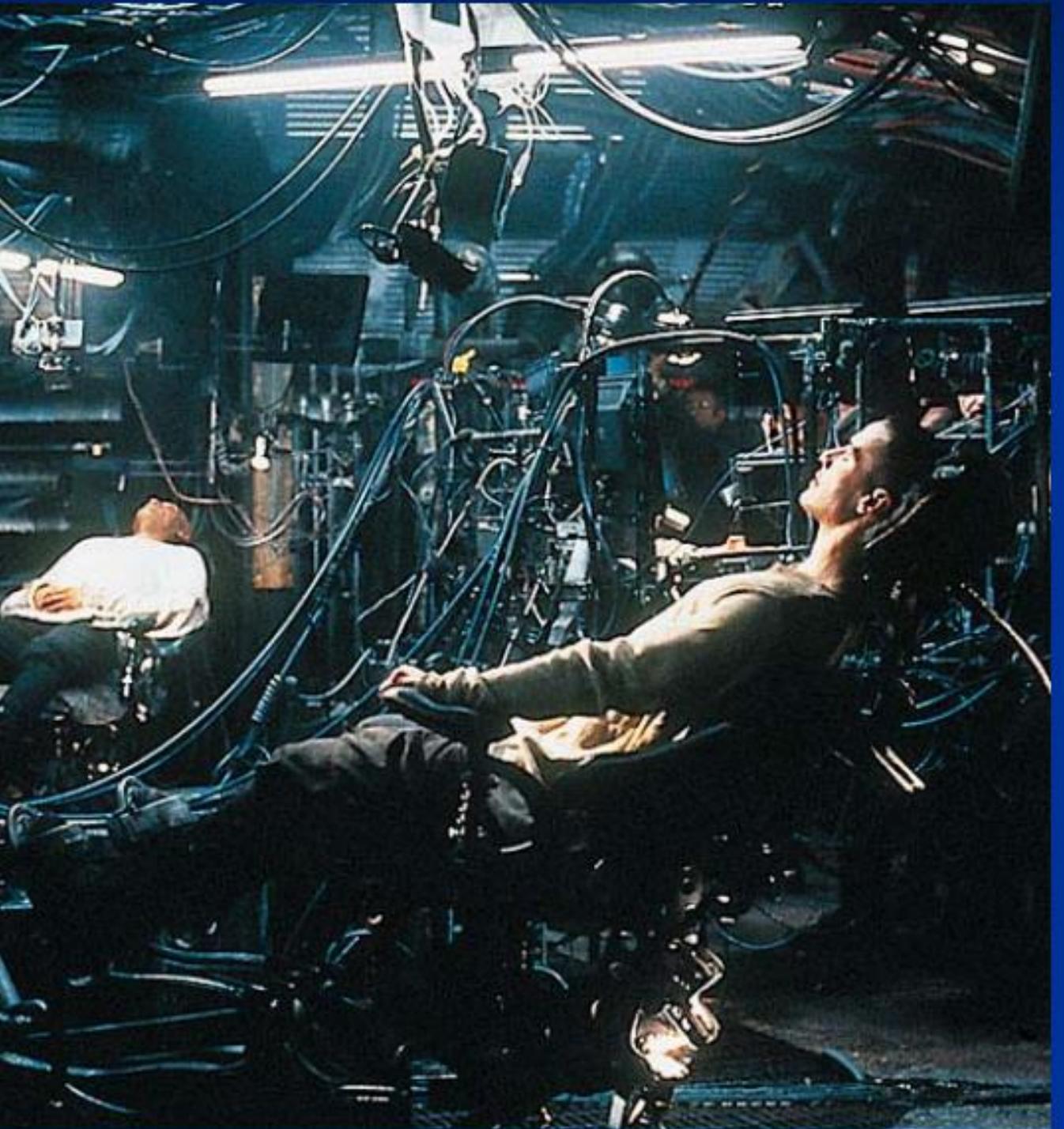
Screenplay
by ROBERT STITZEL and

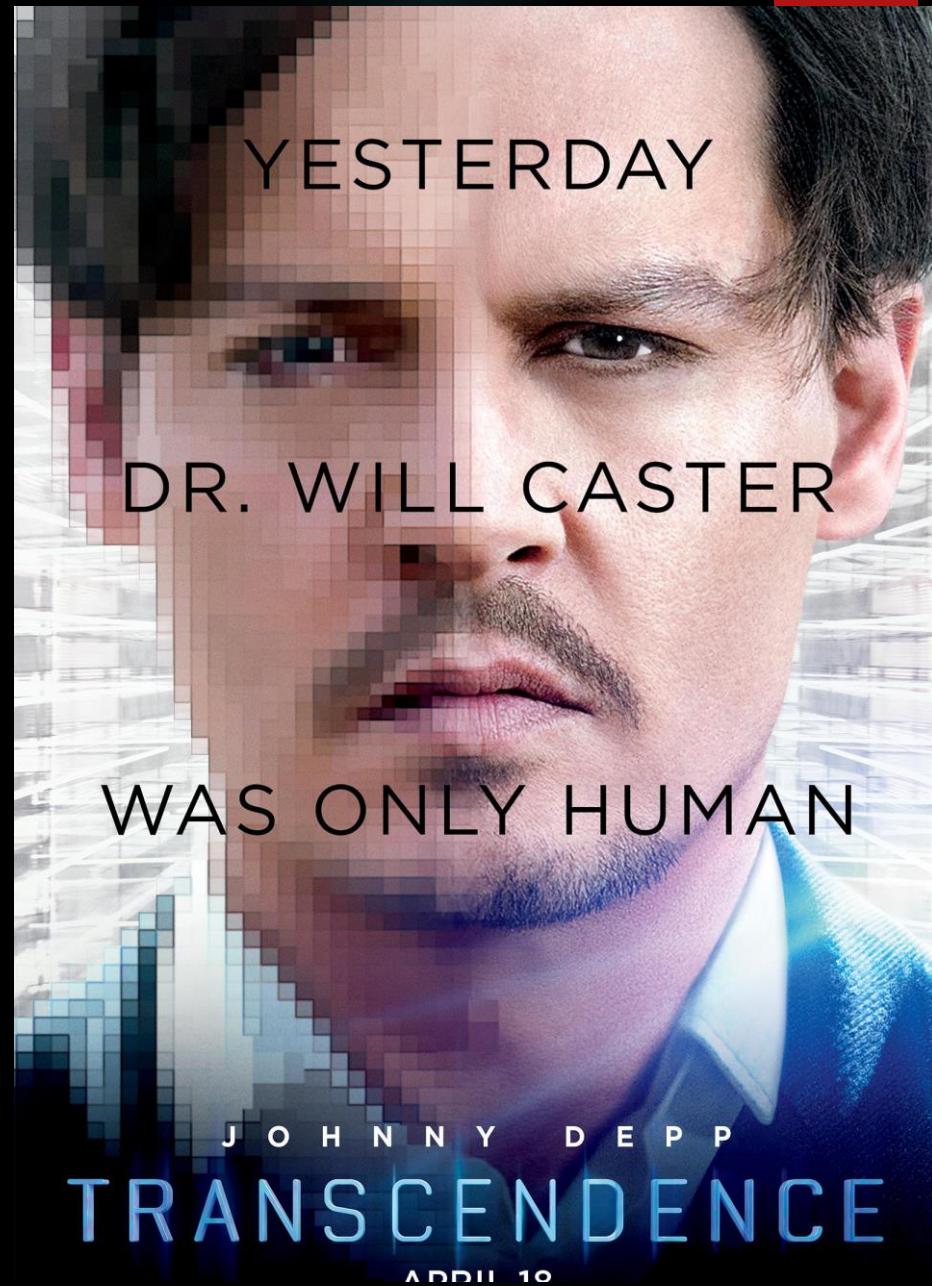
PHILIP FRANK MESSINA
Story by BRUCE JOEL RUBIN Music by JAMES HORNER

Director of Photography RICHARD YURICICH, A.S.C.

Executive in Charge of Production JACK GROSSBERG

Executive Joel L. Freedman







Is it FICTION
or is it
POSSIBLE???

Syllabus

Unit – 1

*Introduction to
Brain Computer
Interface*

Unit – 2

*Introduction to
Basic
Neuroscience*

Unit – 3 *Modelling
and Recoding of
the Brain Signals*

Unit – 4 *Signal
processing*

Unit – 5 *Signal
Analysis using
Machine Learning
Approaches*

Unit – 6 *BCI
Applications*

Course Assessment

- Mid Sem
- End Sem
- Quizzes
 - Surprise Quiz
 - Scheduled Quiz
- Projects
 - Knowledge of Python or MATLAB required

Motivation for BCIs

Potential for restoring lost sensory and motor function

To control prosthetic devices such as prosthetic arms or legs for amputees and patients with spinal-cord injuries

Wheelchairs for paralyzed individuals

Cursors and word spellers for communication by locked-in patients

Sensory prosthetic devices such as cochlear implant for the deaf, retinal implant for the blind

More recently, researchers have begun exploring BCIs for able-bodied individuals for a host of applications such as Game, Entertainment to robotic avatars, biometric identification and Education.

Users



Novice user



Language illiterate



Old-age people



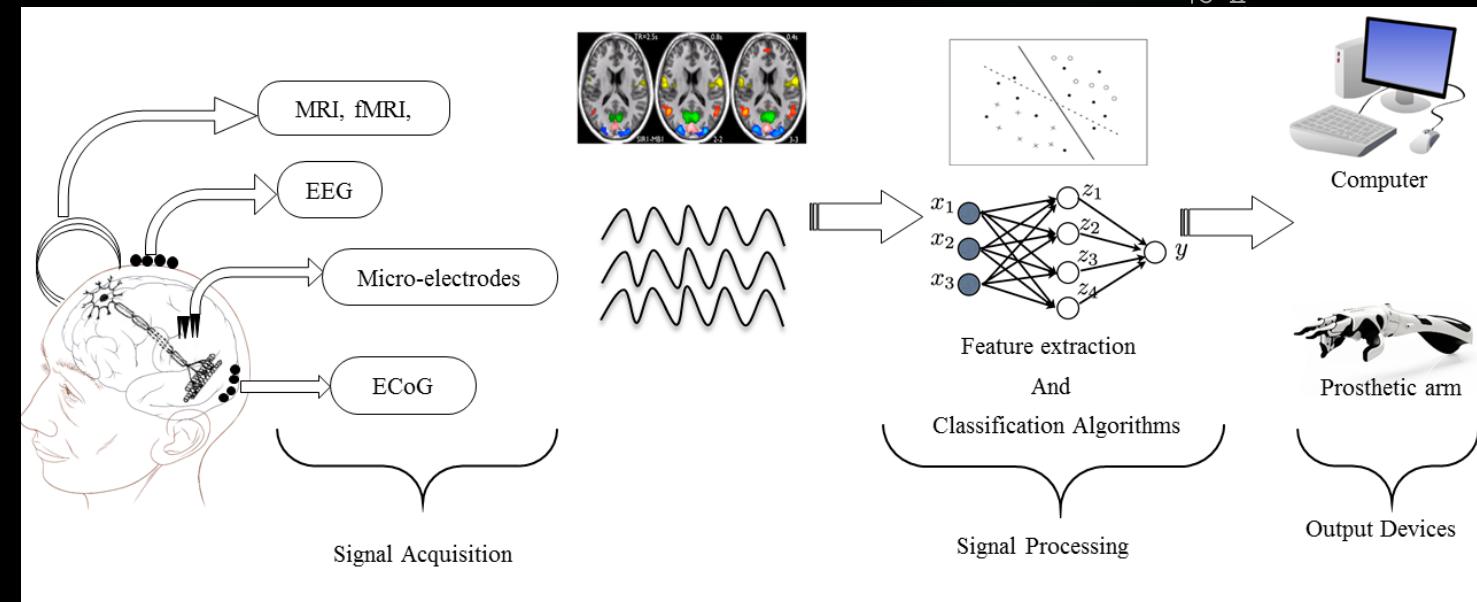
Disabled user



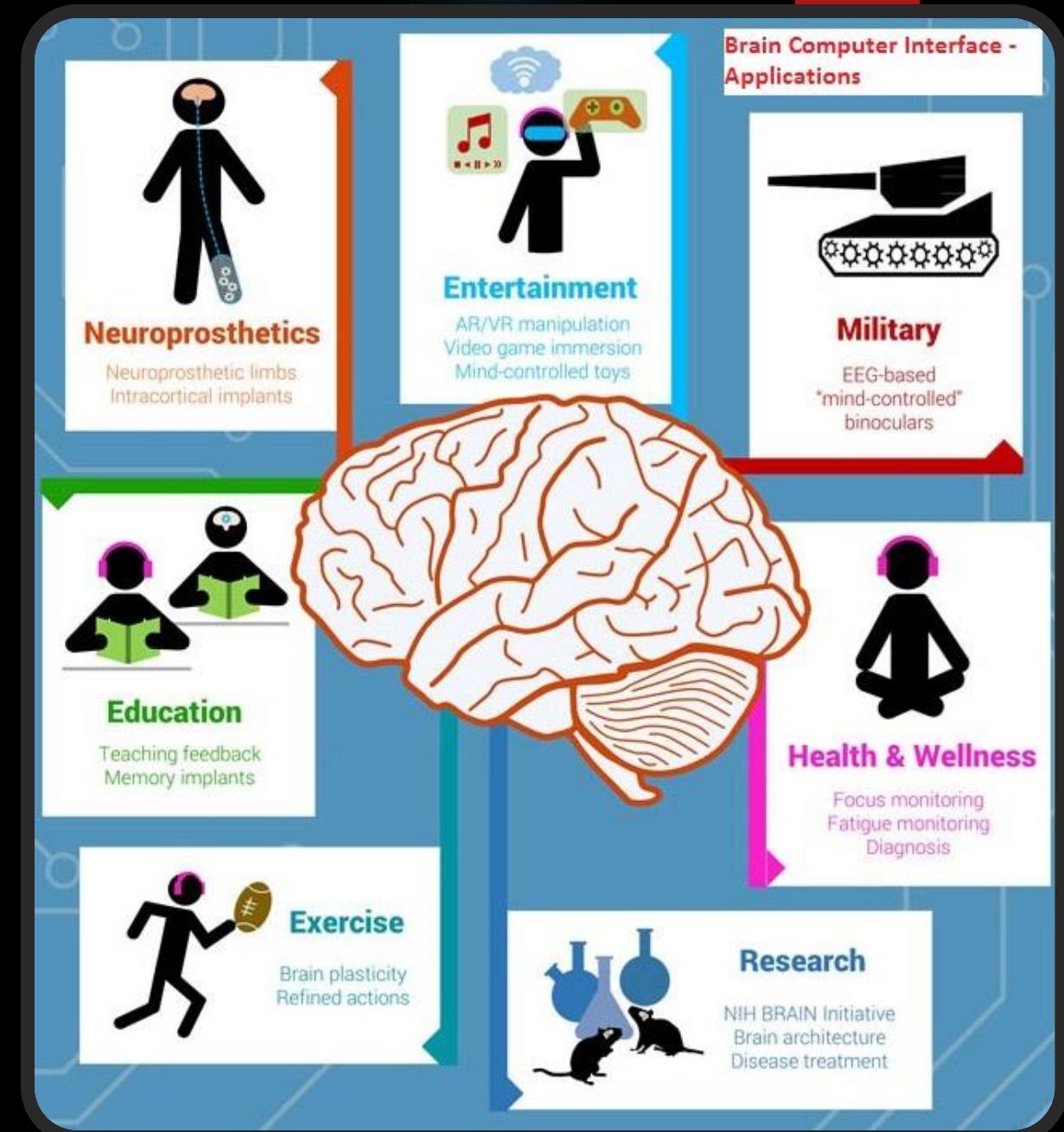
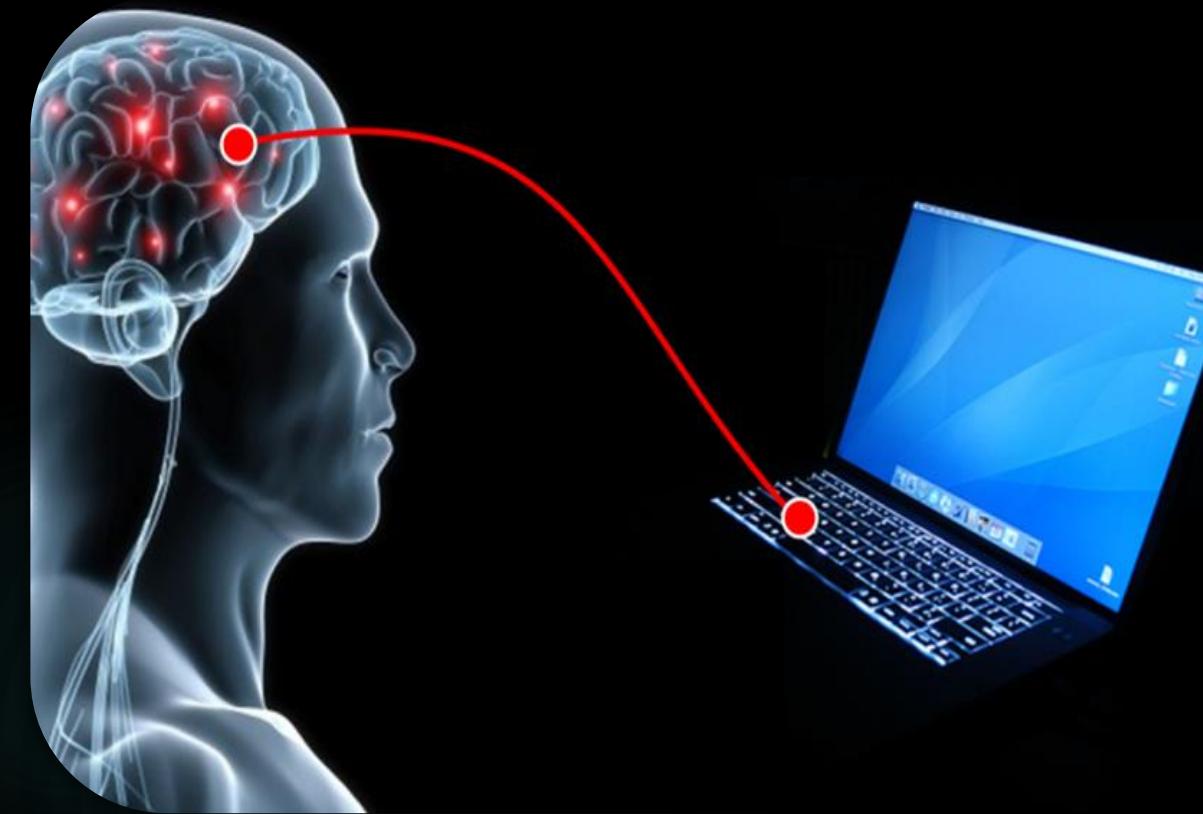
Blind user

Brain Computer Interface

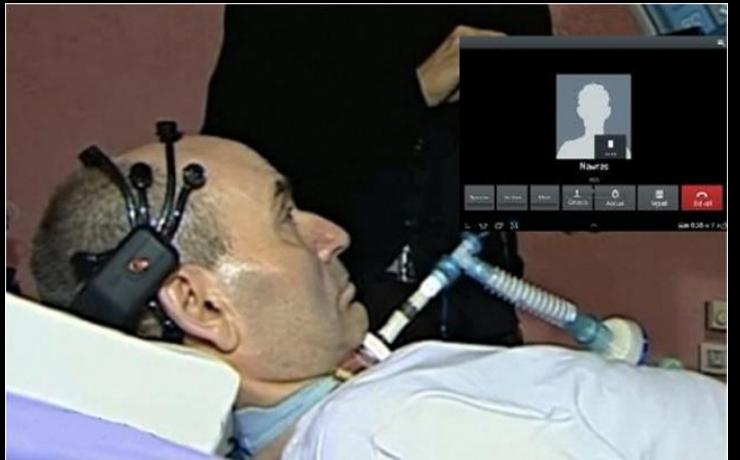
- ▶ A system which translates thoughts and provides an interface used for communication called as Brain Computer Interface (BCI)
- ▶ A typical BCI system comprises of signal acquisition system, signal processing (feature extraction and classification) and an output device



Brain Computer Interaction



Some real-time BCI Applications



Communication



Device Control



Automatic Motion Controlling



Attention Monitoring



Games & Entertainment

Whether BCIs will eventually become as commonplace as current human accessories for sensory and motor augmentation, such as cellular phones and automobiles.????

That remains to be seen.

There are several moral and ethical challenges that society will need to address



History of Brain Computer Interaction

Timeline of BCI

1875

Electrical impulses from a living brain of a rabbit and monkey were recorded for the first time by Richard Caton

1913

Napoleon Cybulski studied the flow of electric current in muscles using his own capacitor.

1969

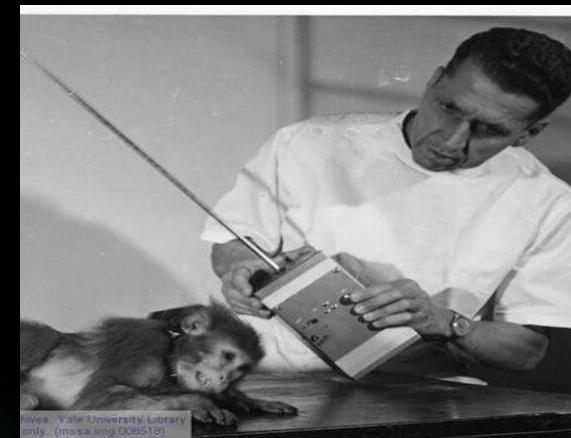
Delgado developed an implantable chip (which he called a “stimoceiver”) that could be used to both stimulate the brain by radio and send electrical signals of brain activity by telemetry, allowing the subject to move about freely.

Adolf Beck studied the brain activity of animals in response to sensory simulation

1890

Hans Berger recorded EEG signals from the lesion area of the human scalp for the first time using a Siemens double-coil galvanometer and non-polarized electrodes.

1920



**1969**

Dr. Eberhard Fetz showed that neural activity could be used to drive an external device.

1990

Philip Kennedy had in 1990 developed “invasive” human brain-computer interface, wires inside the brain attached to a computer.

2004

In 2004, Jonathan Wolpaw and researchers at New York State Department of Health’s Wadsworth Center demonstrated the ability to control a computer using a BCI.

Vidal in 1973 explored the use of scalp-recorded brain signals in humans to implement a simple noninvasive BCI based on “visually evoked potentials”

1973

John Donoghue and his team of Brown University in 2001, commercially design a brain computer interface, the so-called BrainGate.

2001

Phil Kennedy implanted electrodes into his brain in order to establish a connection between his motor cortex and a computer

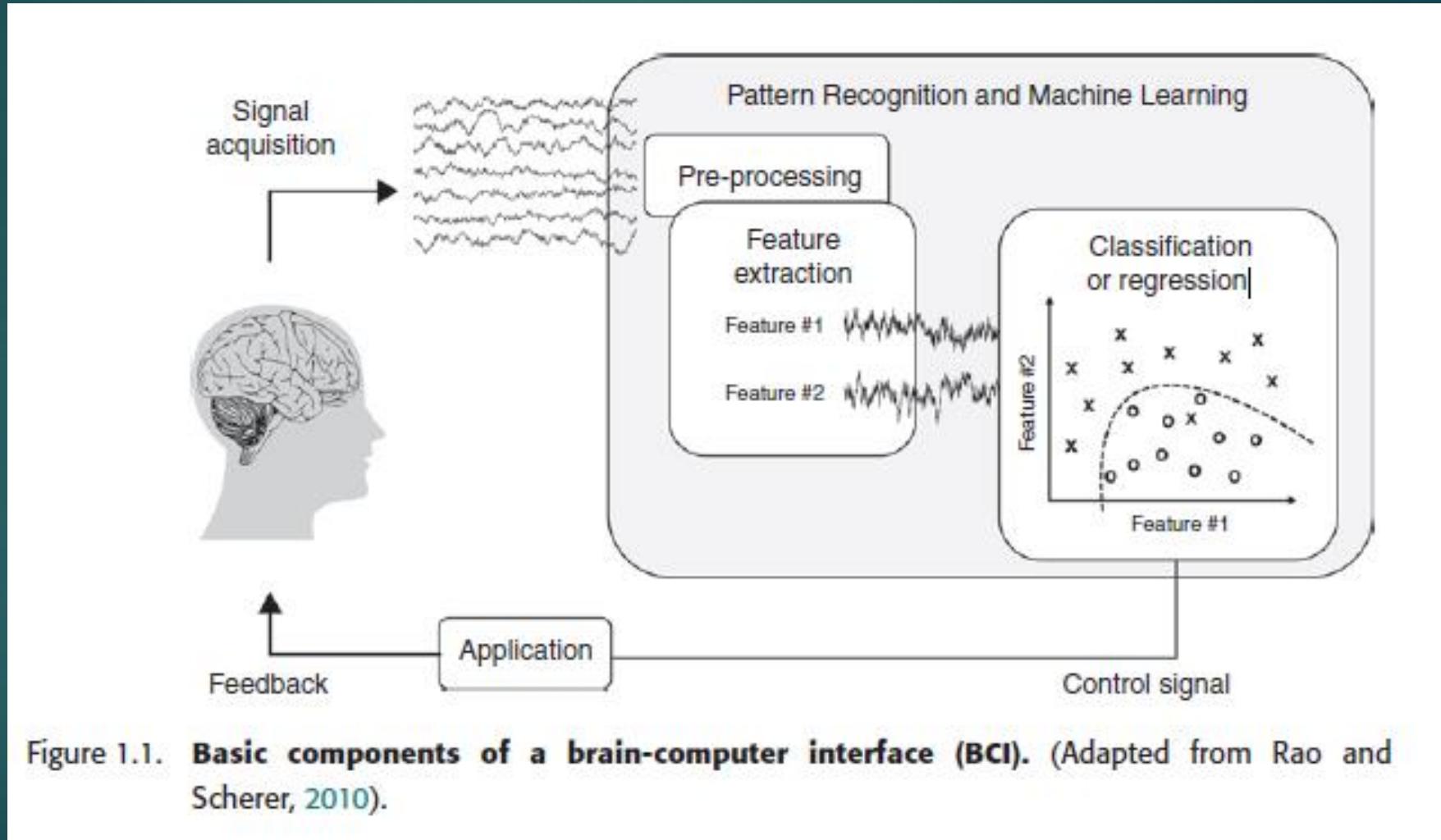
June 2014

To Study the Brain, a Doctor Puts Himself Under the Knife

How one of the inventors of brain-computer interfaces ended up getting one himself.

- ▶ More recently, researchers have begun exploring BCIs for able-bodied individuals for a host of applications such as
 - ▶ Games
 - ▶ Entertainment to robotic avatars, biometric identification,
 - ▶ and Education.

Our Goal in this course



BCI Applications



Hype Cycle for Emerging Technologies, 2020



BCI Applications

- ▶ Device Control
- ▶ User State Monitoring
- ▶ Training and Education
- ▶ Games and Entertainment.
- ▶ Cognitive improvement
- ▶ Safety and Security

Medical Applications



Device controls

- ▶ For rehabilitation
 - ▶ Prosthetic arm
 - ▶ Prosthetic legs

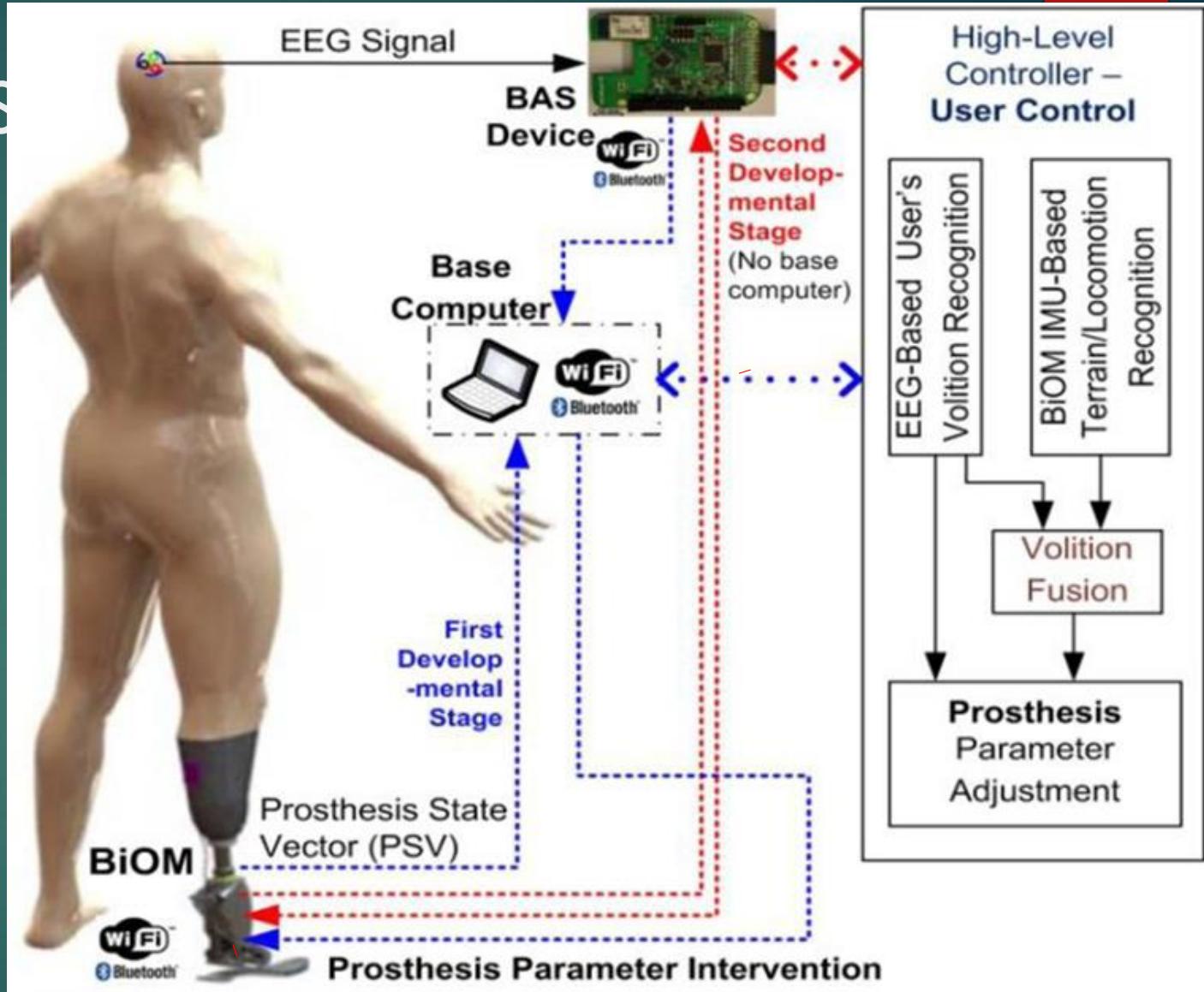
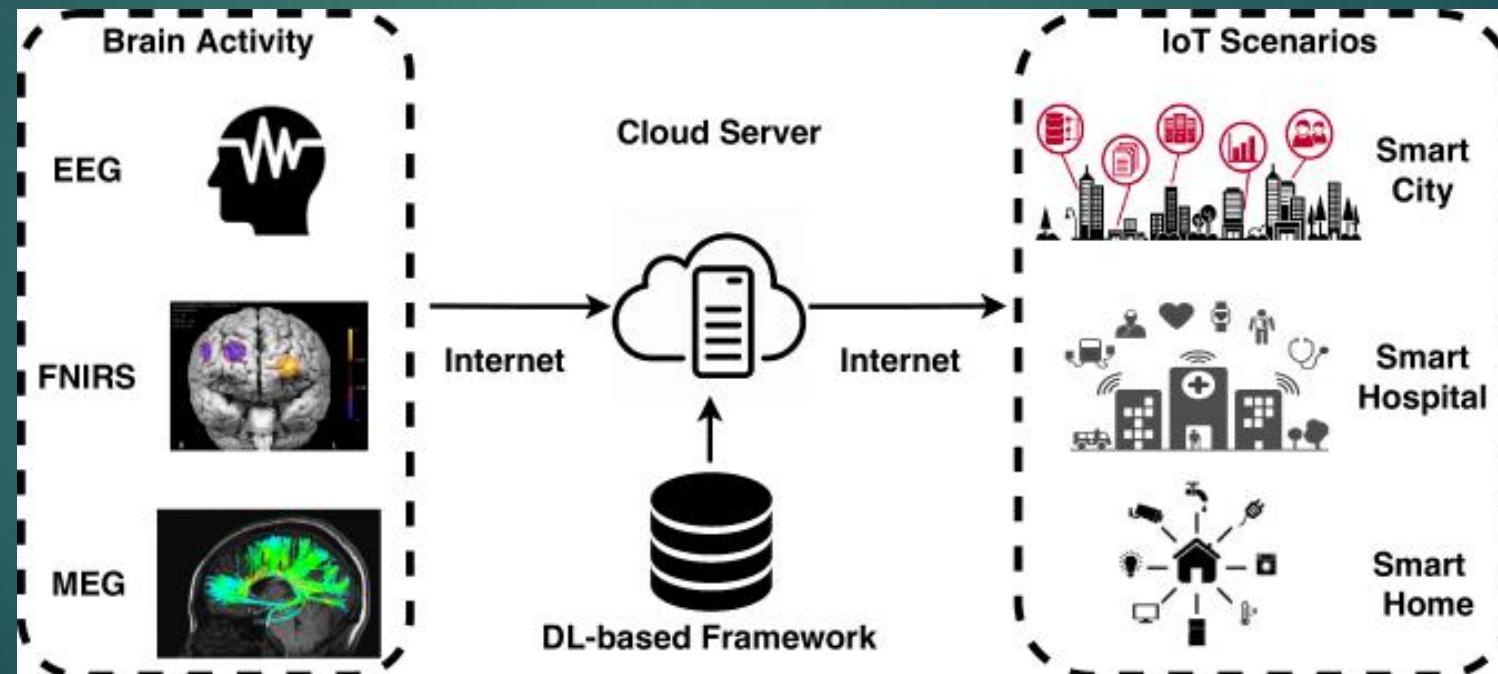


Image credits:

https://www.frontiersin.org/files/Articles/308916/fneur-08-00696-HTML/image_m/fneur-08-00696-g001.jpg

Neuroergonomics and smart environment

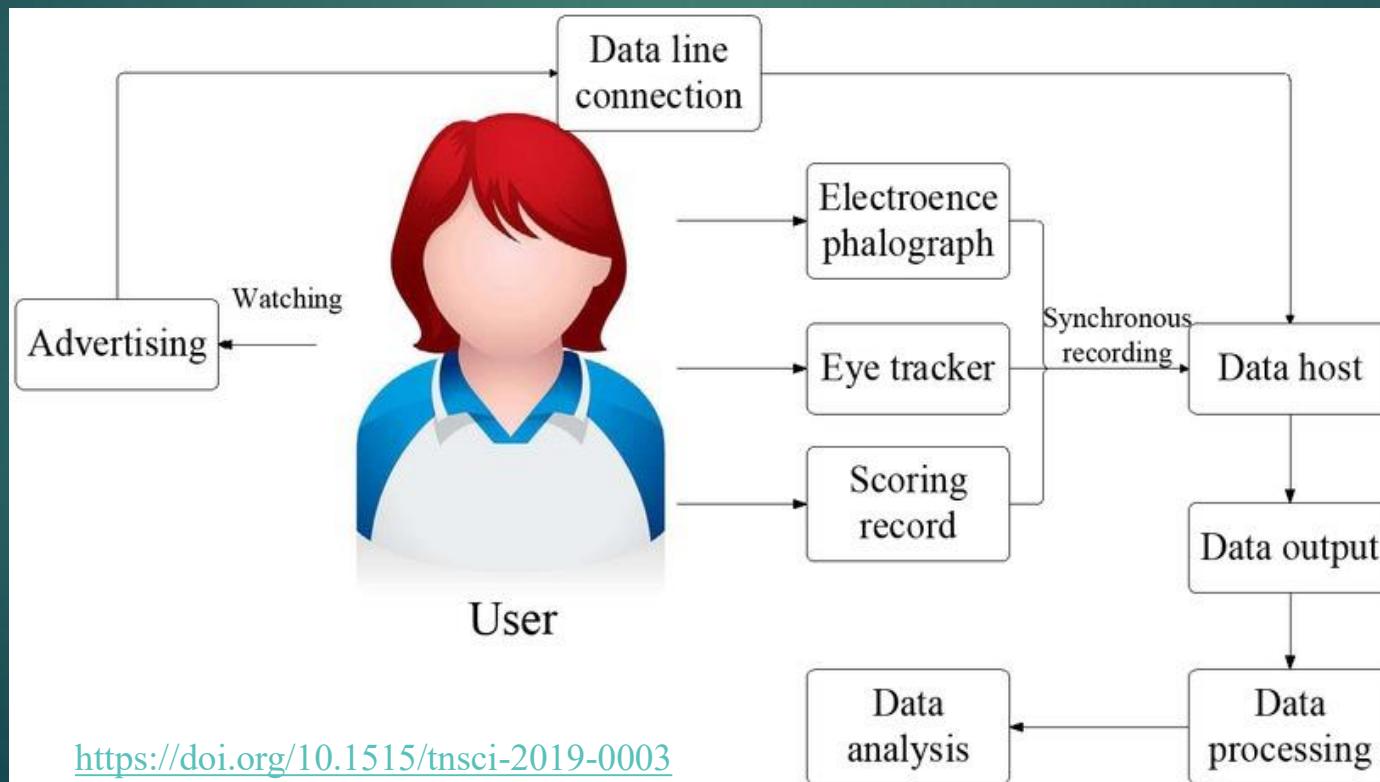
- ▶ Cooperation between Internet of Things (IOT) and BCI technologies
- ▶ intelligent transportation



by [Xiang Zhang, et al.](#) ·

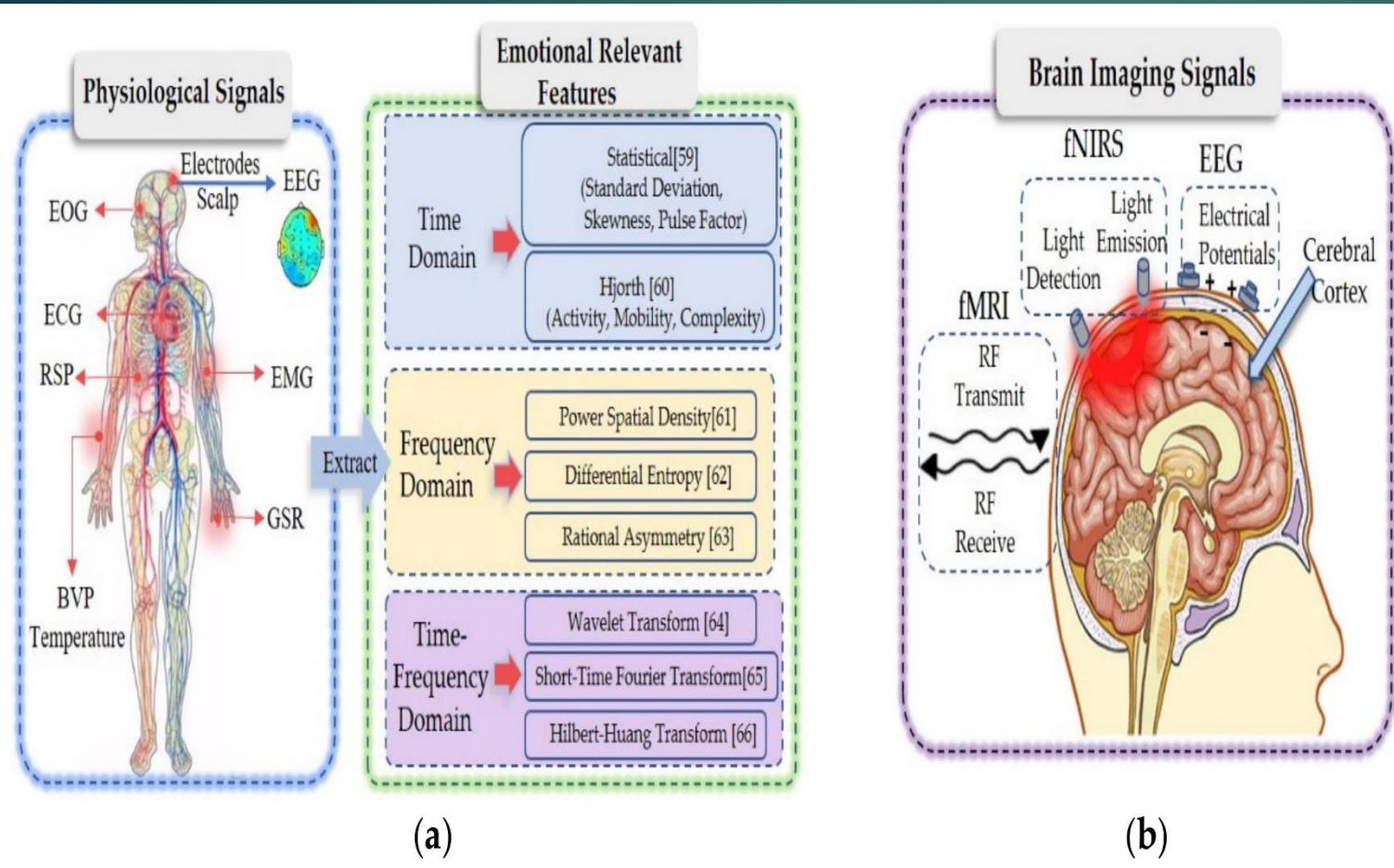
Neuromarketing and advertisement

- ▶ EEG evaluation for TV advertisements related to both commercial and political fields.
 - ▶ The generated attention accompanying watching activity
 - ▶ Estimating the memorization of TV advertisements



Educational and self-regulation

- ▶ Emotional regulation
 - ▶ Use of fMRI–EEG BCI to fight the depression feeling as well as other neuropsychiatric disorders



Advances in Multimodal Emotion
Recognition Based on Brain–Computer
Interfaces
by Zhipeng He

Challenges

- ▶ Usability
- ▶ Hardware
- ▶ Signal processing
- ▶ System integration
- ▶ Cost